Linear Regression

$$y = m * x + c$$

Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable.

MSE(Mean square error):

Mean Square Error (MSE) is defined as Mean or Average of the square of the difference between actual and estimated values.

MAE(Mean absolute error):

Mean Absolute Error (MAE) is the sum of the absolute difference between actual and predicted values.

In [370…

```
import pandas as pd
import numpy as np
```

In [371…

```
data={
    "x" : list(range(1,8)),
    'y' : [1.5,3.8,6.7,9.0,11.2,13.6,16]
}
```

In [372…

```
data
```

Out[372…

```
{'x': [1, 2, 3, 4, 5, 6, 7], 'y': [1.5, 3.8, 6.7, 9.0, 11.2, 13.6, 16]}
```

In [373…

```
dataFrame = pd.DataFrame(data=data)
```

In [374…

```
dataFrame
```

Out[374…

	x	y
0	1	1.5

	x	y
1	2	3.8
2	3	6.7
3	4	9.0
4	5	11.2
5	6	13.6
6	7	16.0

In [375…

```
dataFrame['Sum_xy'] = dataFrame['x'] * dataFrame['y']
```

In [376…

```
dataFrame['sqr_x'] = dataFrame['x']** 2
```

In [377…

```
dataFrame
```

Out[377…

	x	y	Sum_xy	sqr_x
0	1	1.5	1.5	1
1	2	3.8	7.6	4
2	3	6.7	20.1	9
3	4	9.0	36.0	16
4	5	11.2	56.0	25
5	6	13.6	81.6	36
6	7	16.0	112.0	49

Finding M

$$m = \frac{(n * \sum_{i=0}^{n} X_i * Y_i) - (\sum_{i=0}^{n} X * \sum_{i=0}^{n} Y)}{(n * \sum_{i=0}^{n} X^2) - (\sum_{i=0}^{n} X)^2}$$

Finding b

$$b = \frac{\sum_{i=0}^{n} Y_i - m * \sum_{i=0}^{n} X_i}{n}$$

Y-pred

$$Y = m * x + c$$

MSE(Mean Square Error)

$$SSE = \frac{1}{n}\sum_{i=0}^{n}(y_{org} - y_{pred})^2$$

MEA(Mean Absolute Error):

$$mae = \frac{\sum_{i=0}^{n} |y_{org} - y_{pre}|}{n}$$

In [383…

```
class linear_Regression:
    def __init__(self,dataFrame):
        self.n = len(dataFrame)
        self.sum_x = dataFrame['x'].sum()
        self.sum_y = dataFrame['y'].sum()
        self.Sum_xy = dataFrame['Sum_xy'].sum()
        self.sqr_x = dataFrame['sqr_x'].sum()
        self.sumx_h_2 = sum_x ** 2


    def m_val(self,n, sum_x, sum_y, Sum_xy, sqr_x, sumx_h_2):   ## Find m value
        self.num_m = n*((Sum_xy))-(sum_x)*(sum_y)
        self.den_m = n*((sqr_x))-(sumx_h_2)
        self.m = self.num_m / self.den_m
        return self.m

    def b_val(self,n, sum_x, sum_y):       ## Find b value
        self.num_b = (sum_y) - m*(sum_x)
        self.den_b = n
        self.b = self.num_b / self.den_b
        return self.b

    def fit_train(self,m,b,dataFrame):      # y-pred(y = m*x + c)
        self.y_pre = [(m*x_val) + b for x_val in dataFrame['x']]
        return self.y_pre

    def mse_val(self,dataFrame,y_pre):   # Mse
        diff1 = []
        for yorg, ypred in zip(dataFrame['y'],y_pre):
            diff = (yorg - ypred)**2
            diff1.append(diff)
        sse = sum(diff1)
        mse=(1/n)*(sse)
        return mse

    def mae_val(self,dataFrame,y_pre):    # Mae
        diff1 = []
        for yorg, ypred in zip(dataFrame['y'],y_pre):
            diff= abs(yorg - ypred)
            diff1.append(diff)
        ae = sum(diff1)
        mae = ae/n
        return mae
```

In [384…

```
m_obj = linear_Regression(dataFrame)    ## object declaration for m(slope)
m_obj.m_val(n, sum_x, sum_y, Sum_xy, sqr_x, sumx_h_2)   # obj.methodname()
print("m value:",m)                                     # print m value
```

```
m value: 0.024081632653061246
```

In [385…

```
b_obj = linear_Regression(dataFrame)   ## object declaration for b(constant)
b_obj.b_val(n, sum_x, sum_y            # obj.methodname()
print("b value:",b)                    # print b value
```

```
b value: -0.8285714285714231
```

In [386…

```
ypre_obj = linear_Regression(dataFrame)     ## object declaration for y-predictions
ypre_obj.fit_train(m,b,dataFrame)           # obj.methodname()
print("y_pre value:",y_pre)                 # print y-pred values
```

```
y_pre value: [1.5857142857142903, 4.0000000000000036, 6.414285714285717, 8.8285714285714
3, 11.242857142857142, 13.657142857142857, 16.07142857142857]
```

In [387…

```
mse_obj=linear_Regression(dataFrame)        ## object daclaration for Mean square error
m=mse_obj.mse_val(dataFrame,y_pre)          # obj.methodname()
print("mean square error value:",m)         # print Mean square error value(mse)
```

```
mean square error value: 0.024081632653061246
```

In [388…

```
mae_obj=linear_Regression(dataFrame)        ## object daclaration for Mean absolute err
a=mae_obj.mae_val(dataFrame,y_pre)          # obj.methodname()
print("mean absolute error value:",a)       # print mean absolute error(mae)
```

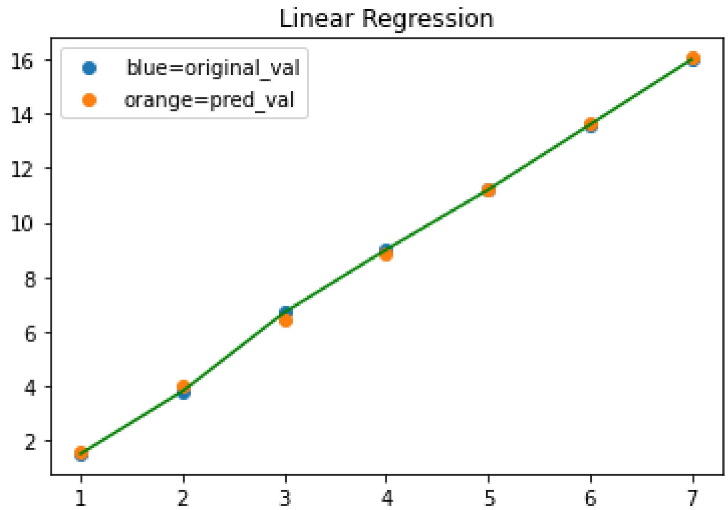
```
mean absolute error value: 0.13061224489795956
```

In [423…

```
#graph
import matplotlib.pyplot as plt

x_val = dataFrame['x']
yorg = dataFrame['y']
ypred = y_pre

plt.plot(x_val,yorg,color='g')
plt.scatter(x_val,yorg,label='blue=original_val')    # for original values
plt.scatter(x_val,ypred,label='orange=pred_val')     # for predicted values
plt.title("Linear Regression")
plt.legend()
plt.show()
```


Linear Regression
blue=original_val
orange=pred_val
16
14
12
10
8
6
4
2
1
2
3
4
5
6
7

```
In [ ]:
```

```
In [ ]:
```