

# Linear Regression

## สมาชิกกลุ่ม

นายกฤตพล	ผ้าเจริญ	6401012620161
นายเจษฎา	ศรีจุลโพธิ์	6401012620170
นายศุภกร	พลศิริ	6401012620234
นายสิรภพ	ห้วงวิไล	6401012630132

1.เขียนโปรแกรมสำหรับสร้างแบบจำลองเชิงเส้นด้วยวิธีลดตามความชัน พร้อมทั้งแสดงฟังก์ชันค่าใช้จ่ำยในรูปของคอนทัวร์และแสดงให้เห็นถึงขั้นตอนในการปรับพารามิเตอร์  
(Lecture หน้าที 49)

**Representation:** Linear Regression

$$h(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_dx_d$$

**Evaluation:** Mean Squared Error: MSE

$$\frac{1}{2n} \sum_{i=1}^n (h(x^{(i)}) - y^{(i)})^2$$

```
def CalculateMSE(x_positions, y_positions, w0, w1):  
    n = len(x_positions)  
    MSE = 0  
    for i in range(0, n):  
        MSE += (w0 + w1*x_positions[i] - y_positions[i]) ** 2  
    MSE *= 1/(2*n)  
    return MSE
```

# Gradient Descent

ทำซ้ำ {

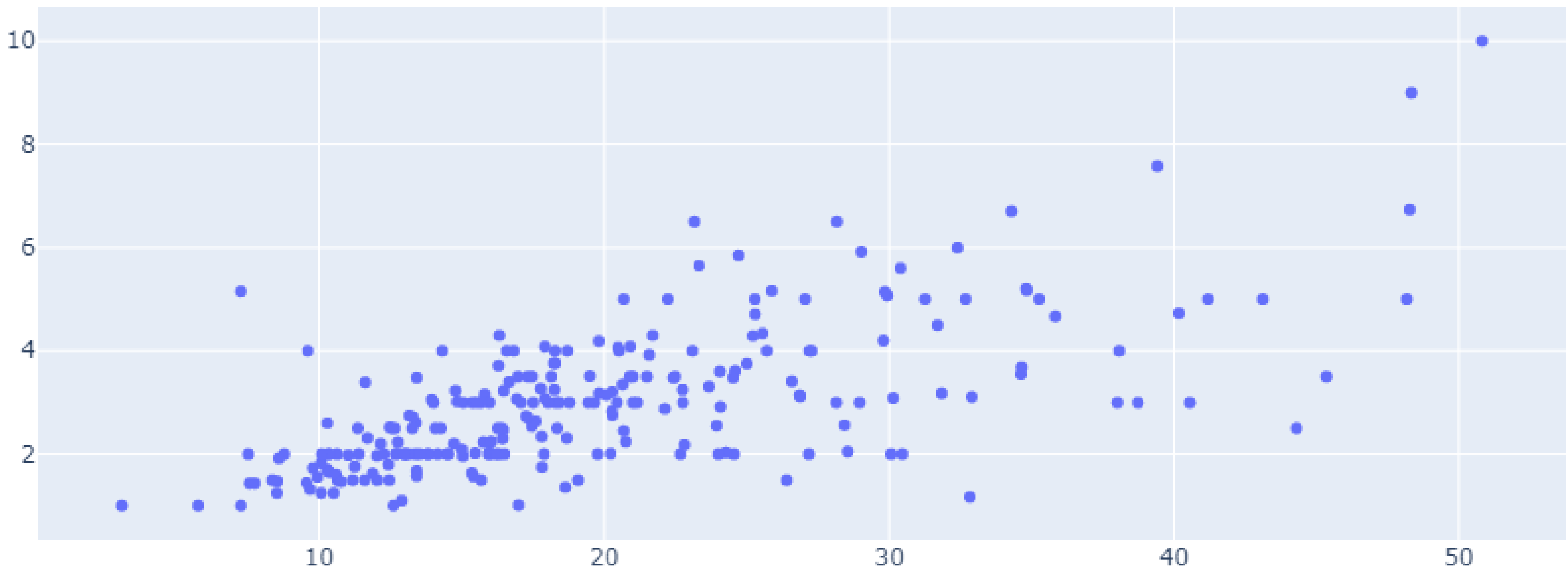
$$w_0 := w_0 - \alpha \frac{1}{n} \sum_{i=1}^n (h(x^{(i)}) - y^{(i)})$$

$$w_1 := w_1 - \alpha \frac{1}{n} \sum_{i=1}^n (h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

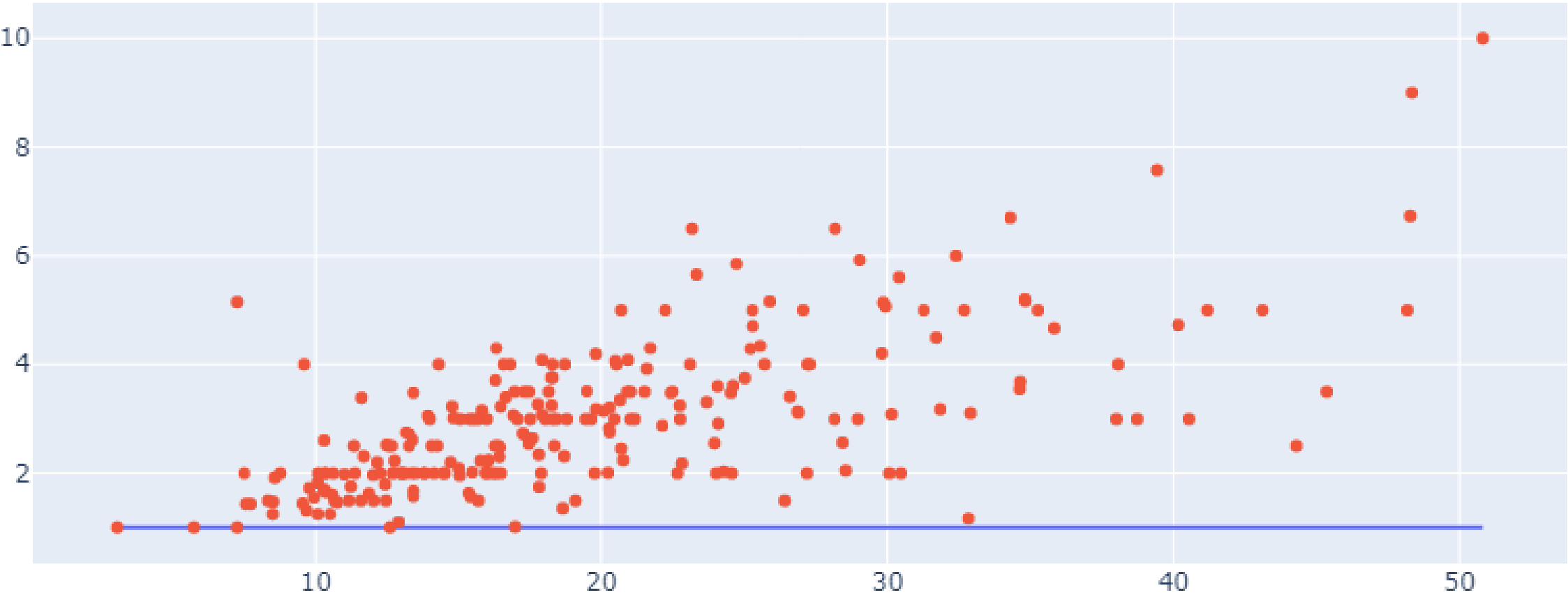
}

```
def GradientDescent(x_positions, y_positions, a, w0, w1):  
    n = len(x_positions)  
    w0_new = w0  
    w1_new = w1  
    for i in range(0,n):  
        w0_new -= (a / n) * (w0 + w1*x_positions[i] - y_positions[i])  
        w1_new -= (a / n) * (w0 + w1*x_positions[i] - y_positions[i]) * x_positions[i]  
    return (w0_new ,w1_new)
```

# ข้อมูลที่นำมาใช้

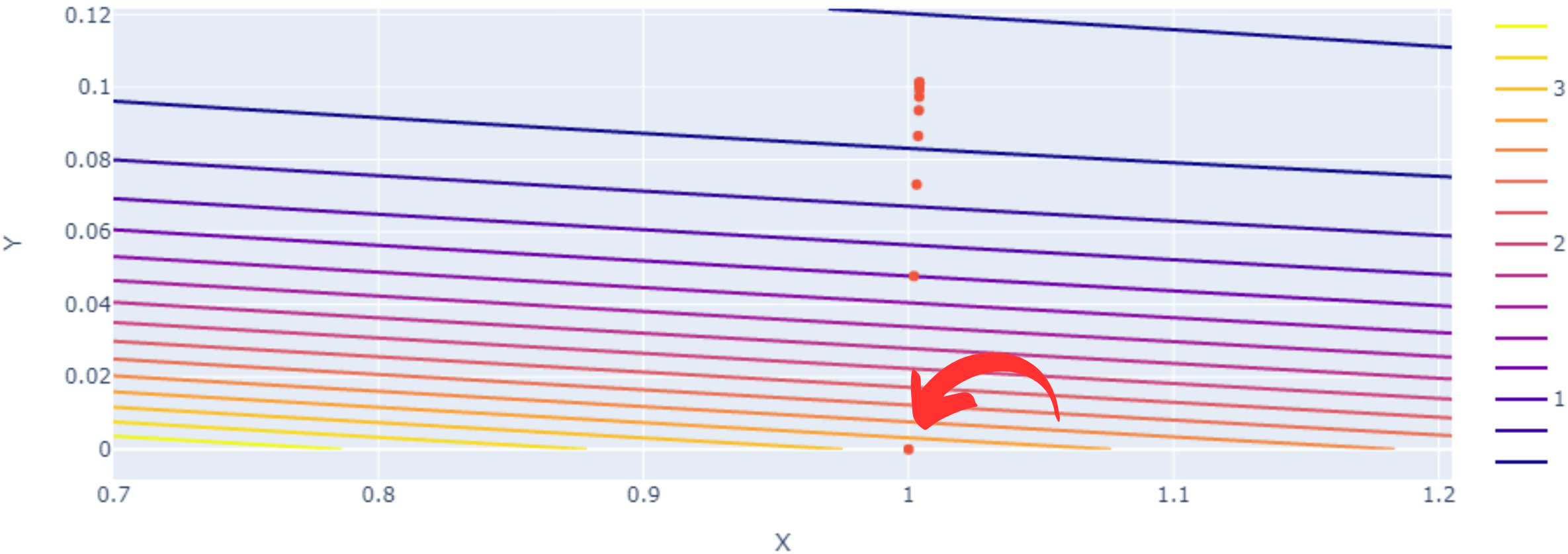


$w_0 = 1, w_1 = 0$  (ค่าที่เลือกตอนเริ่มต้น)

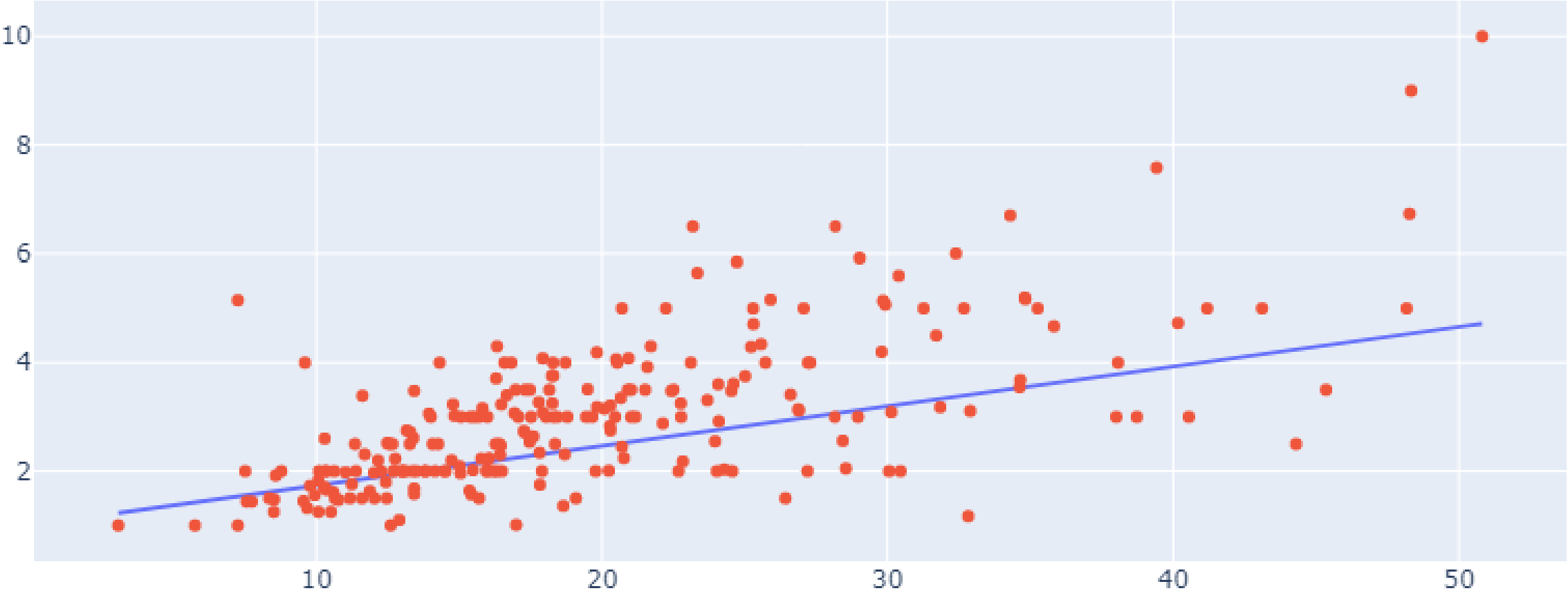


Contour Plot

$E = 2.94986$

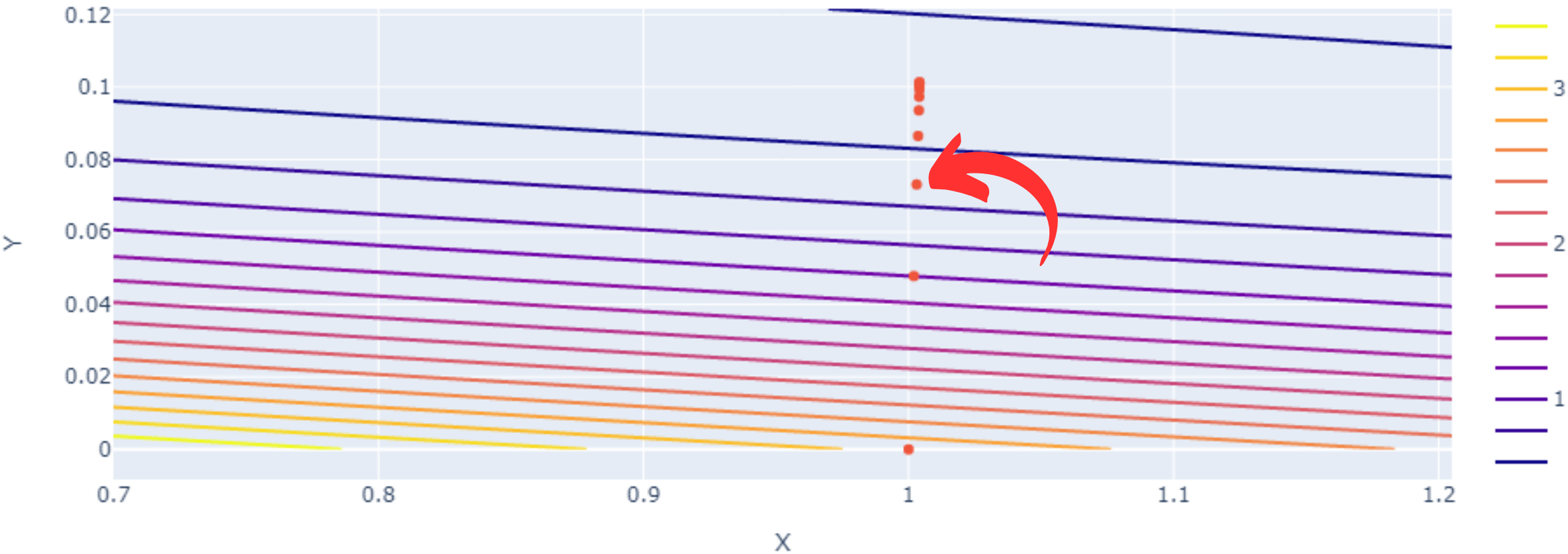


$w_0 = 1.00304, w_1 = 0.07311$



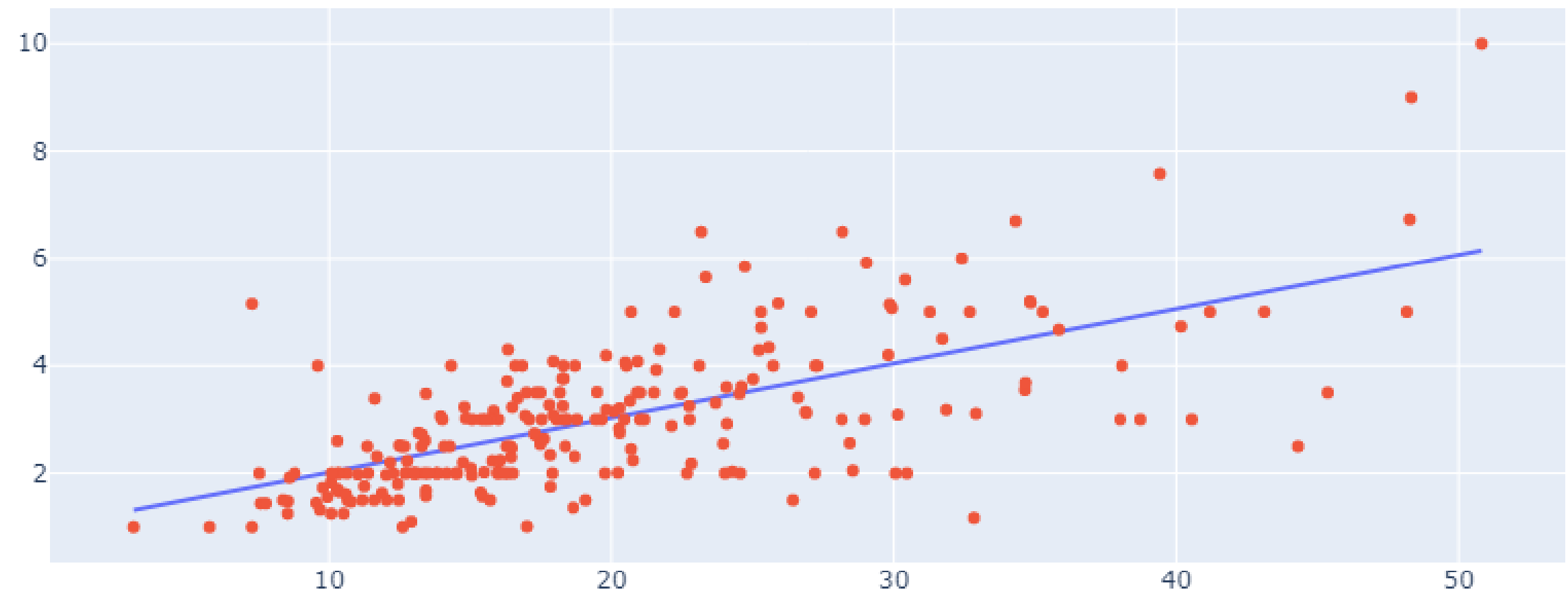
Contour Plot

$E = 0.70864$



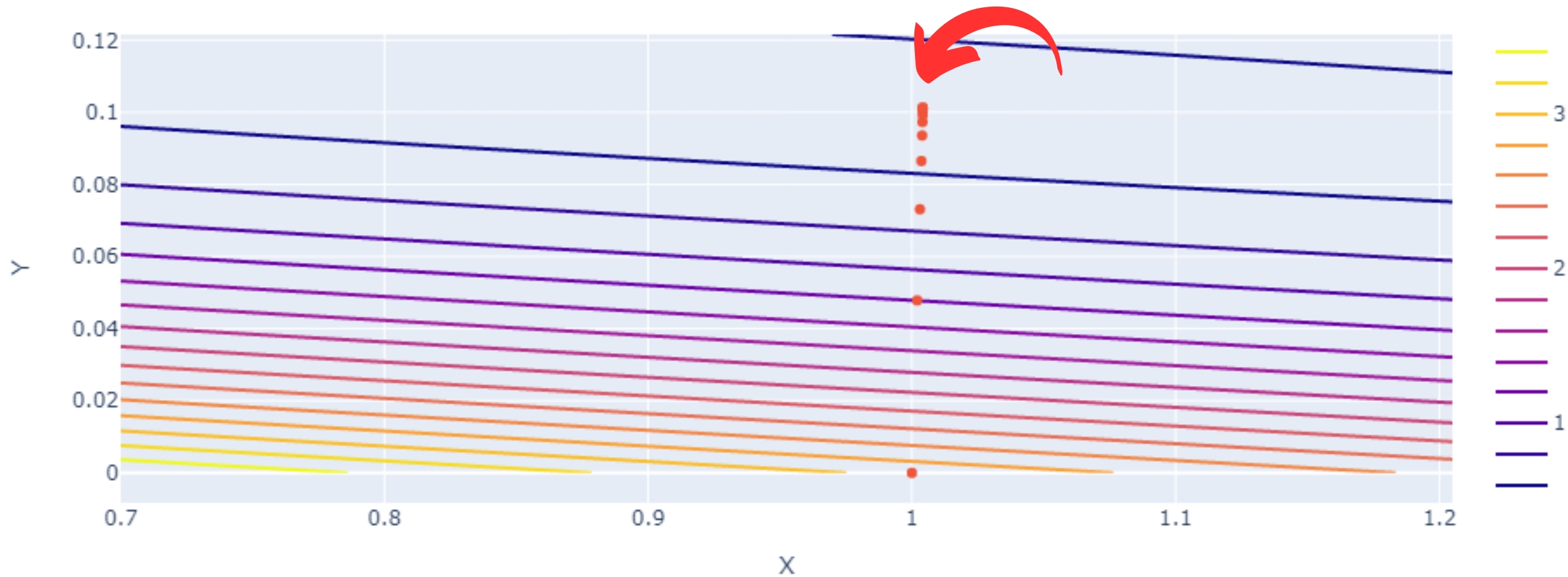


$w_0 = 1.00412$  ,  $w_1 = 0.10132$  (ค่าที่ได้ E น้อยที่สุด)



Contour Plot

$E = 0.5186$



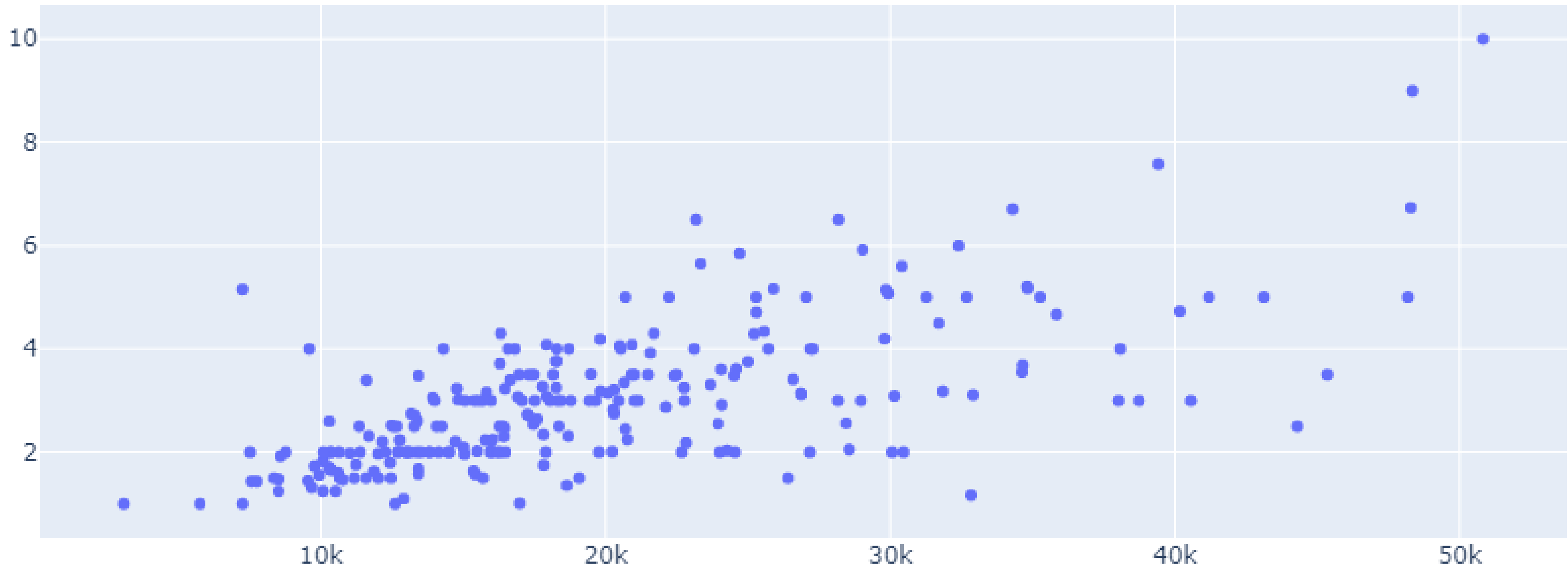
2.เขียนโปรแกรมสำหรับแสดงผลกระทบต่อการทำงานของวิธี  
ลดตามความชันและฟังก์ชันค่าใช้จ่ำย เมื่อตัวแปร  $x$  หลายตัวมี  
ค่าแตกต่างกันมาก และ แสดงผลของการปรับปรุง  
ประสิทธิภาพด้วยการทำให้เป็นมาตรฐาน (Lecture หน้าที่  
63)

$$x_{j,std}^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

$$S_x = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

```
def Standardize(points):  
    a = 0  
    n = len(points)  
    mean = sum(points) / n  
    new_points = []  
    for i in range(0,n):  
        a += (points[i] - mean) ** 2  
    std = math.sqrt(a / (n-1))  
    for i in range(0,n):  
        new_points.append((points[i] - mean) / std)  
    return new_points
```

## ข้อมูลที่นำมาใช้



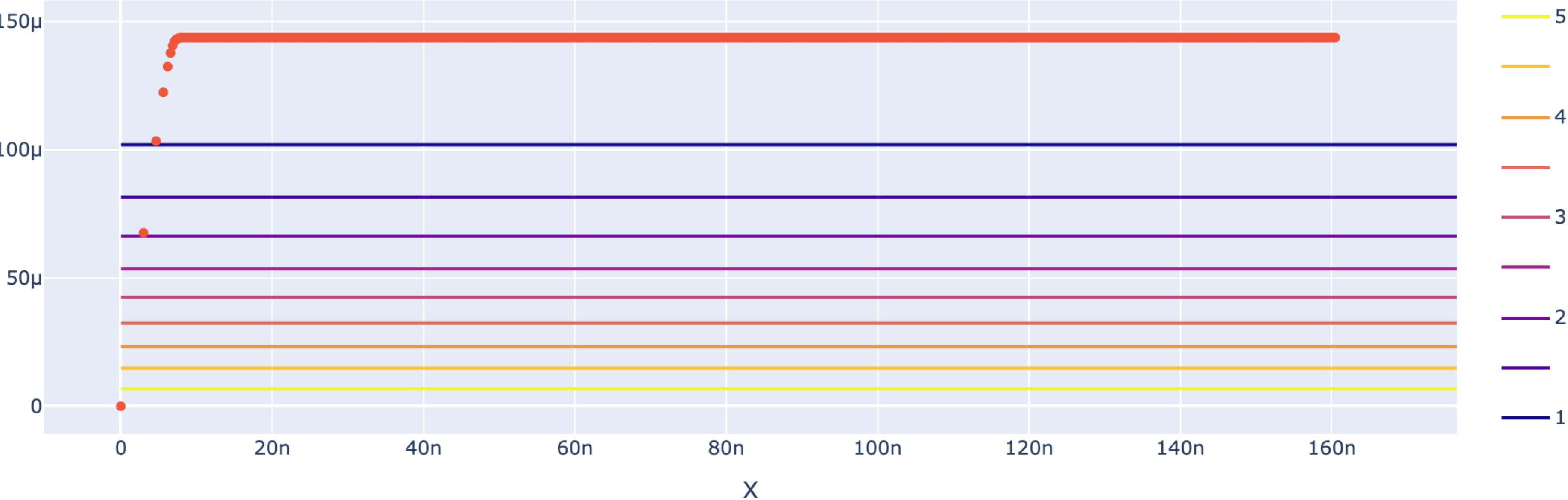
นำค่า x มาคูณ 1000 เพื่อให้ค่าแตกต่างกันมาก

# ก่อนทำ Standardization

Learning Rate =0.0000000001



Contour Plot

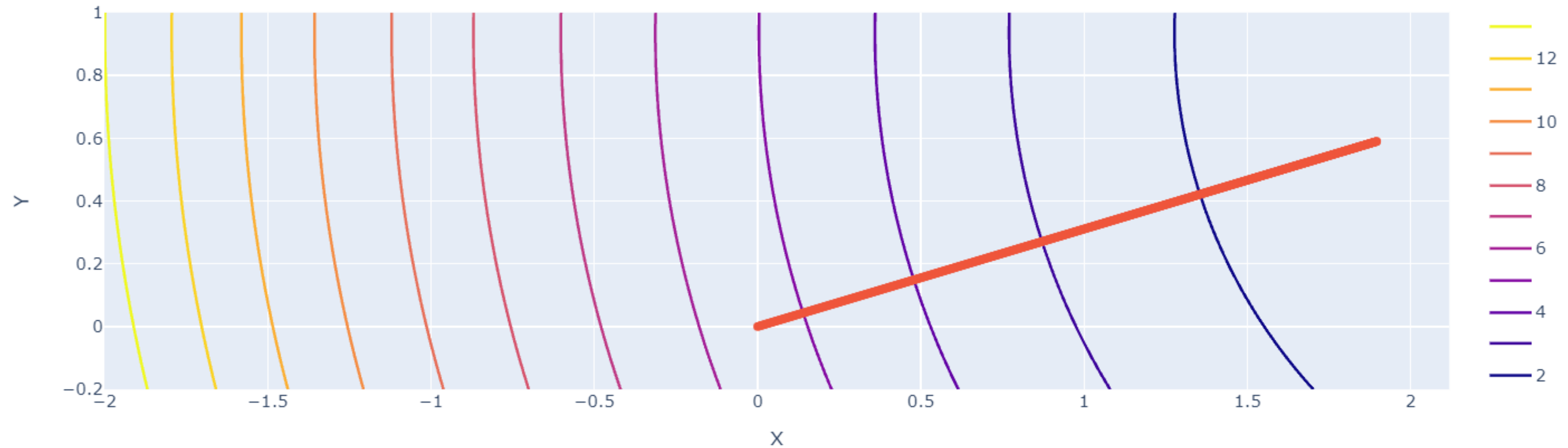


จำนวนครั้งของการปรับเปลี่ยนความชันคือ 1000 ครั้ง

# หลังทำ Standardization

Learning Rate =0.001

Contour Plot



จำนวนครั้งของการปรับเปลี่ยนความชันคือ 1000 ครั้ง

## เปรียบเทียบเวลาที่ใช้ในการ Run code

ก่อนทำ Standardization

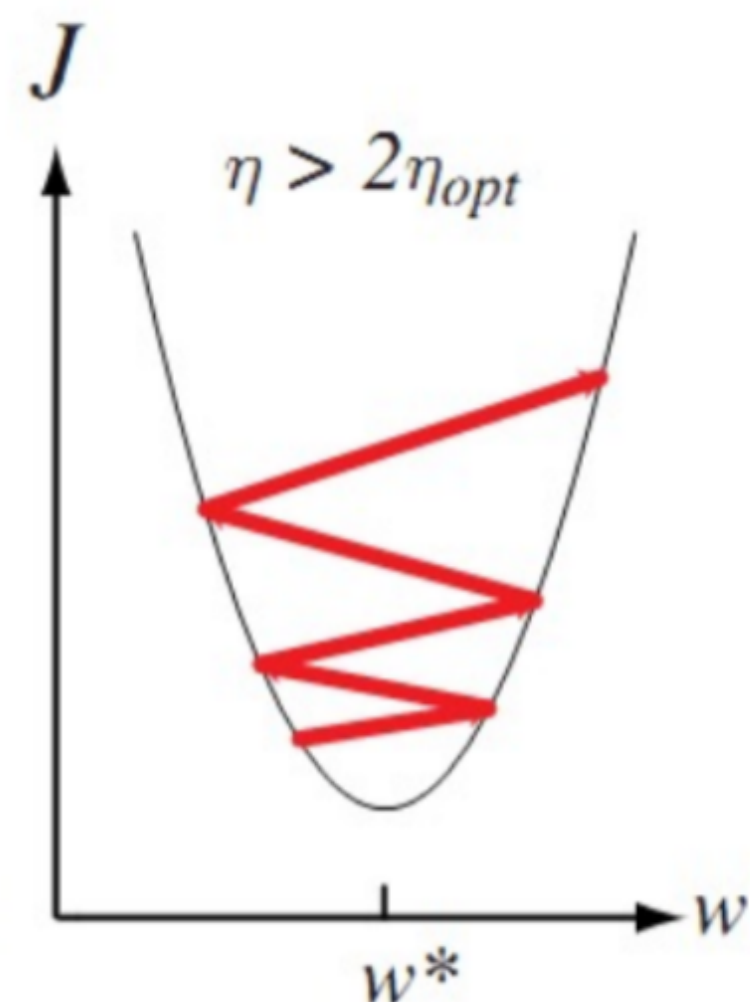
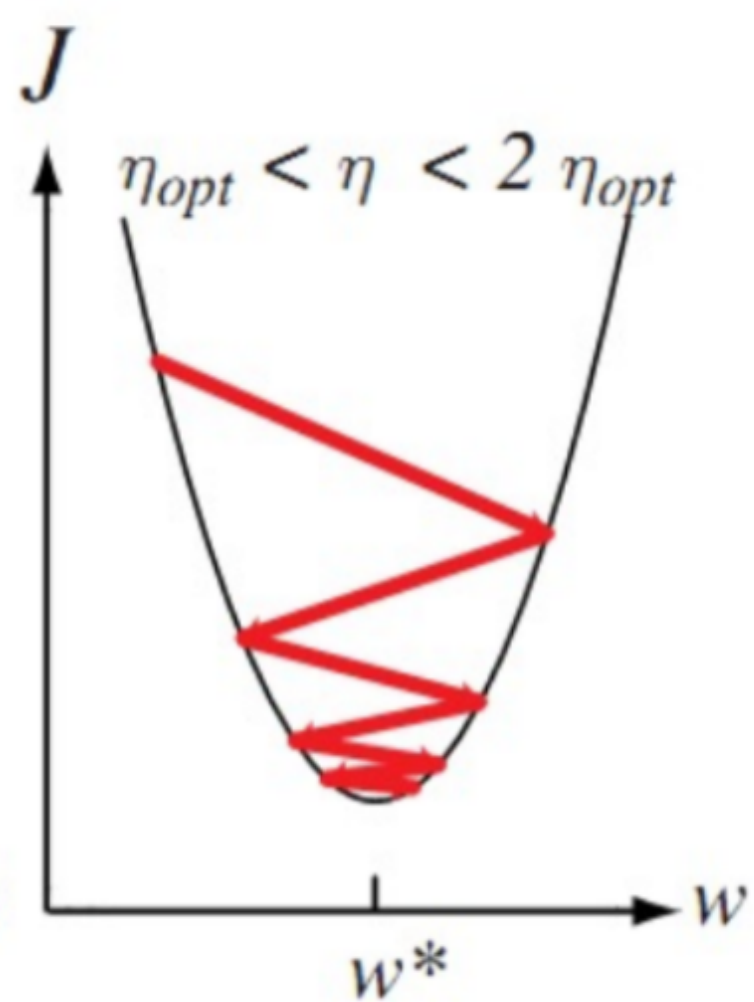
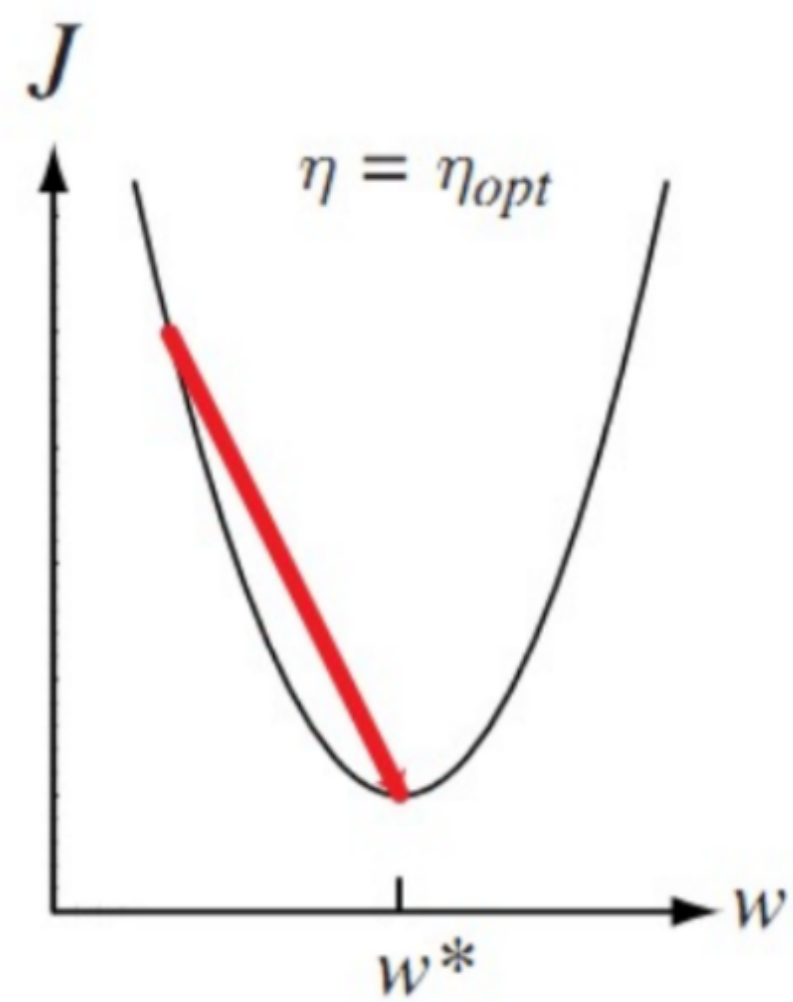
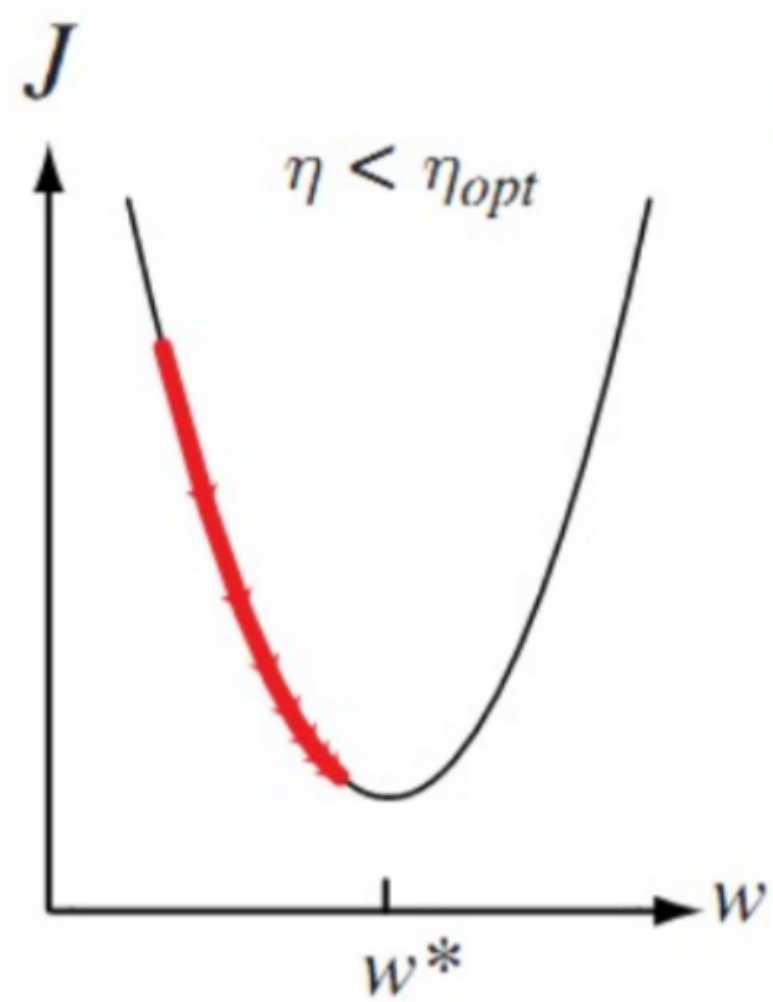
```
9] ✓ 7.1s
. 5.4481418032786895
  1.6045313563512328e-0
  [5.4481418032786895,
  7.175729990005493
```

หลังทำ Standardization

```
[32] ✓ 0.7s
... 0.4979508196721309
    -1.2373192595694802
    [0.4979508196721309
    0.7244181632995605
```

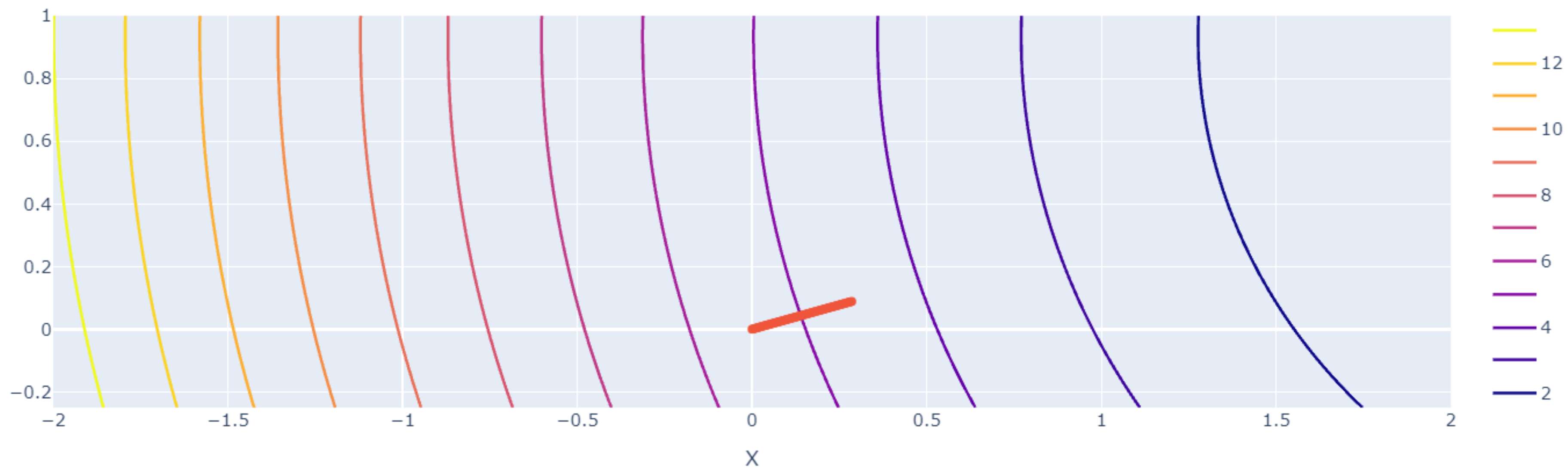
### 3.เขียนโปรแกรมสำหรับแสดงผลของการปรับพารามิเตอร์ การเรียนรู้ (Lecture หน้า ที่ 66)





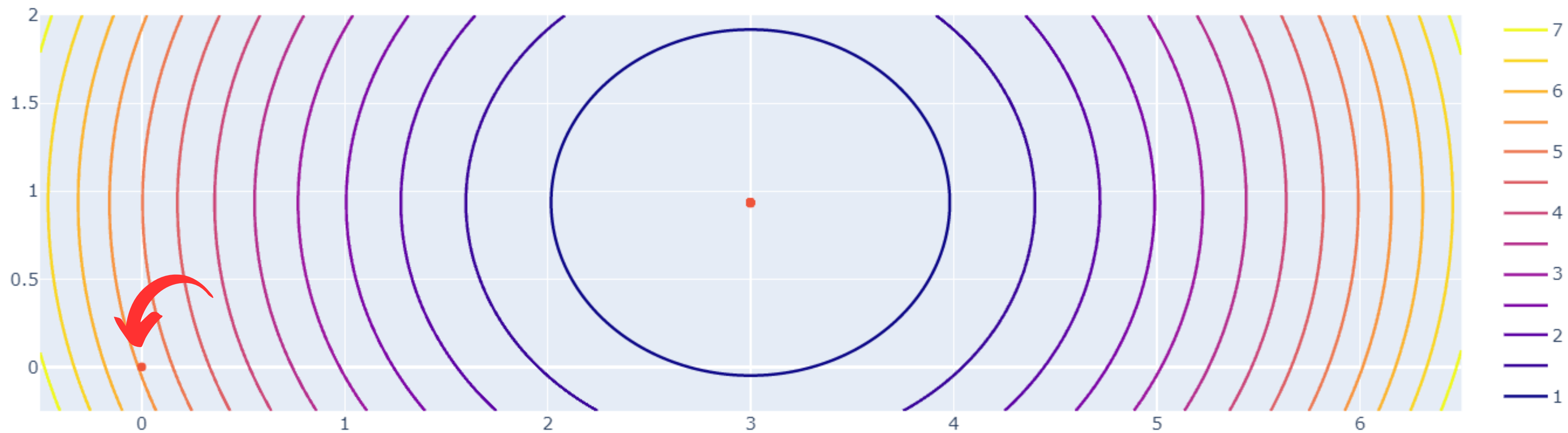
Learning Rate = 0.001

Contour Plot



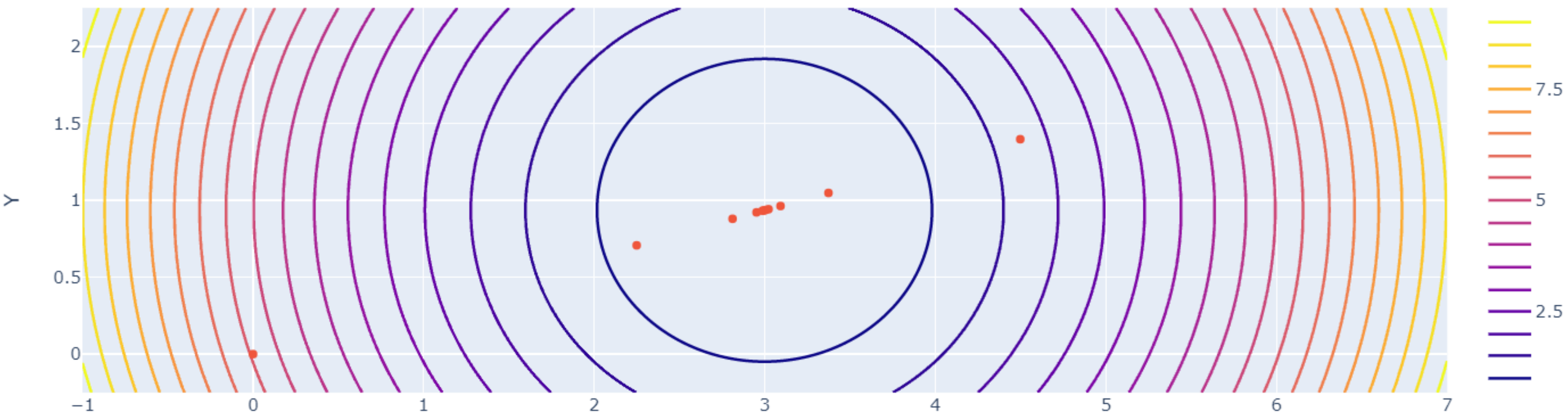
# Learning Rate = 1

Contour Plot



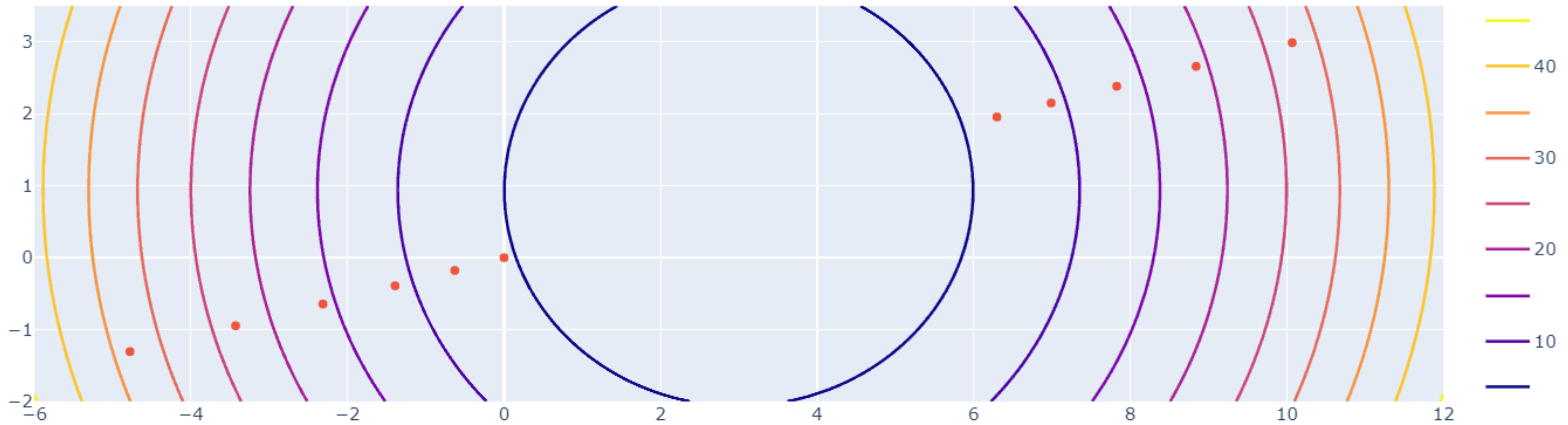
Learning Rate = 1.5

Contour Plot



Learning Rate = 2.1

Contour Plot



4.เขียนโปรแกรมสำหรับเปรียบเทียบผลลัพธ์ที่ได้จากวิธี  
สมการปรกติและวิธีลดตามความชัน (Lecture หน้าที่ 57  
และ 59)

## Normal Equation

```
def normalEquation(matX, vecY):  
    X_transpose = np.transpose(matX)  
  
    left_side = np.dot(X_transpose, matX)  
    right_side = np.dot(X_transpose, vecY)  
  
    ans = np.linalg.inv(left_side).dot(right_side)  
    return ans
```

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

## GradientDescent

```
def gradientDescent(x_position, y_position, learning, ans):  
    n = len(x_position)  
    X_trans = np.transpose(x_position)  
    Xw = np.dot(x_position, ans)  
    temp = np.dot(X_trans, (Xw - y_position))  
    w_new = ans - (learning/n)*temp  
    return w_new
```

$$\mathbf{w} := \mathbf{w} - \frac{\alpha}{n} \mathbf{X}^T (\mathbf{Xw} - \mathbf{y})$$



# Input

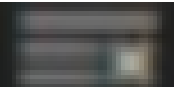
$$X = \begin{pmatrix} 1 & 2015 & 497 \\ 1 & 1584 & 356 \\ 1 & 2469 & 556 \\ 1 & 989 & 222 \end{pmatrix} \quad y = \begin{pmatrix} 400 \\ 300 \\ 500 \\ 200 \end{pmatrix}$$

# ผลลัพธ์ที่ได้

```
Gradient Descent : [350.          134.11044447  -5.4287512 ]
```

```
-----  
Normal Equation : [350.          134.11044447  -5.4287512 ]
```

# Thank you!

```
+ Code + Markdown | ☐ Interrupt    
22 a = 0.00000007  
23 for i in range(300000000):  
24     W = gradientDescent(X,Y,a,W)  
25     if i % 300000000 == 0:  
26         print(W)  
27 print(W)  
28
```