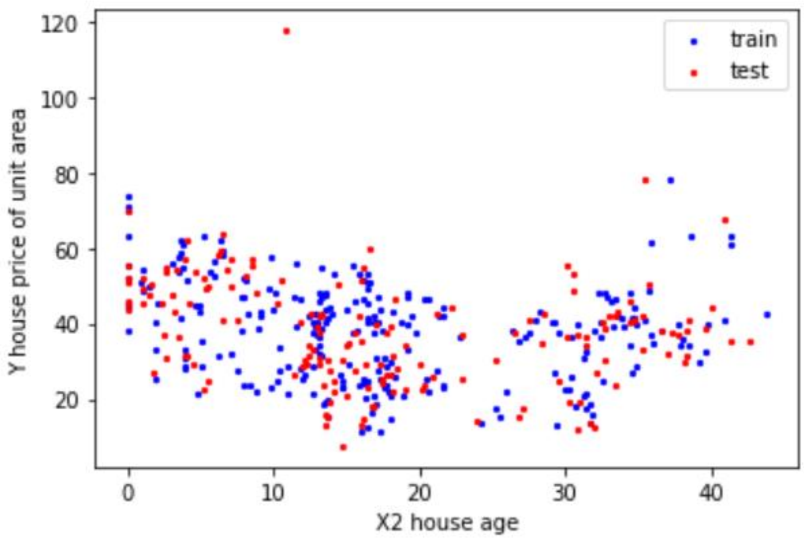
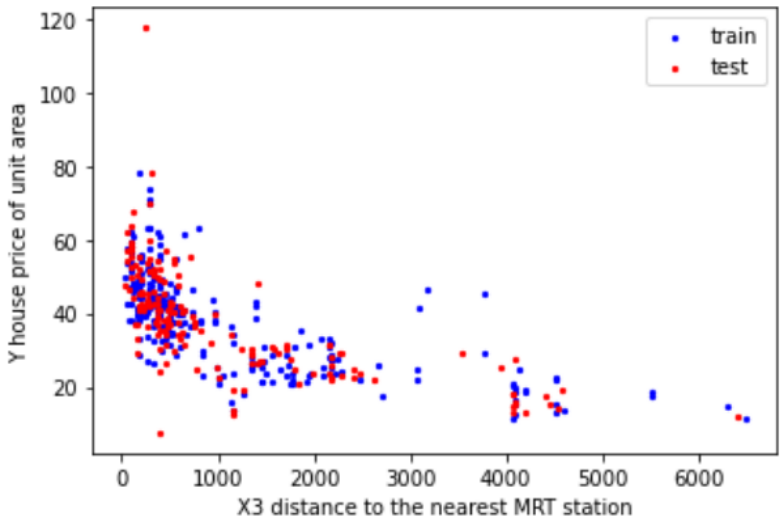
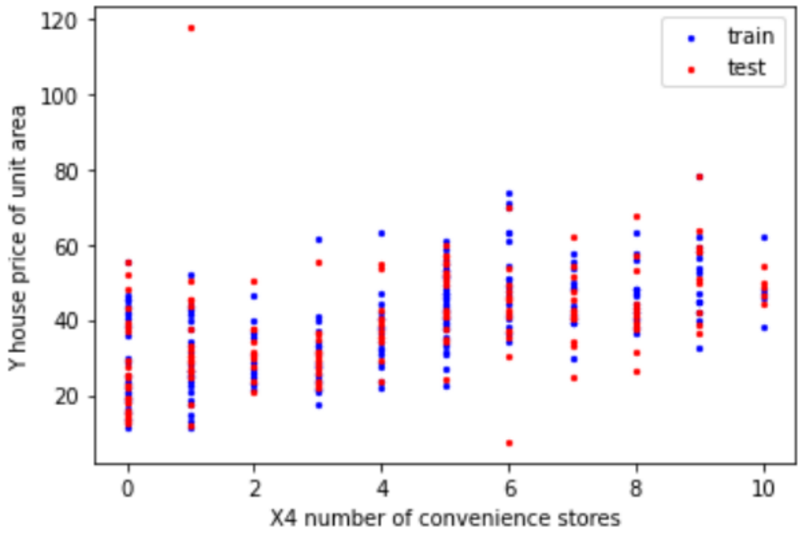
ML HW3 109070032 廖品睿

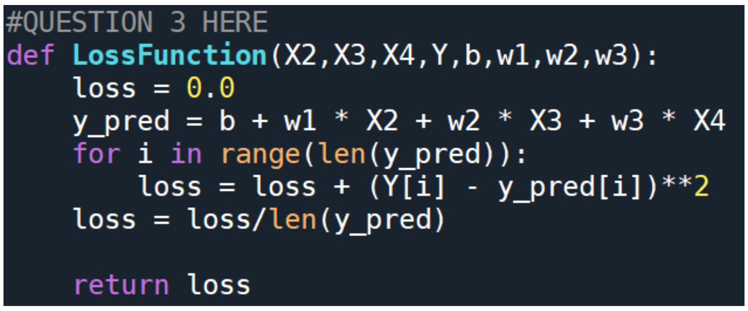
2.







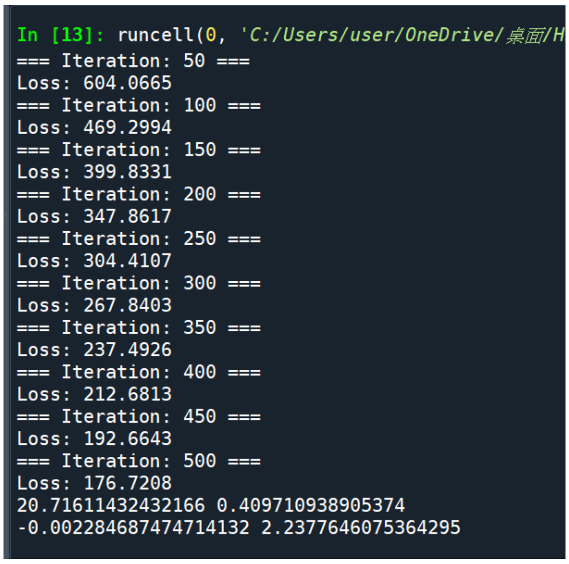
3.



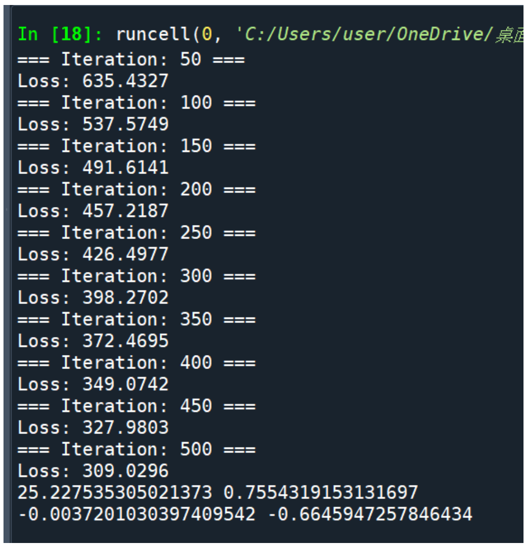
```
#QUESTION 3 HERE
def LossFunction(X2,X3,X4,Y,b,w1,w2,w3):
    loss = 0.0
    y_pred = b + w1 * X2 + w2 * X3 + w3 * X4
    for i in range(len(y_pred)):
        loss = loss + (Y[i] - y_pred[i])**2
    loss = loss/len(y_pred)

    return loss
```

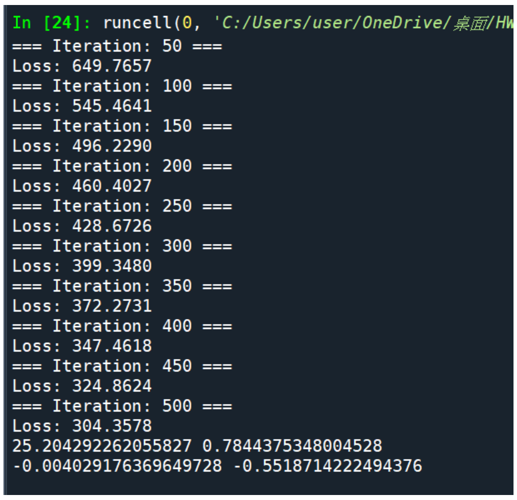
4.



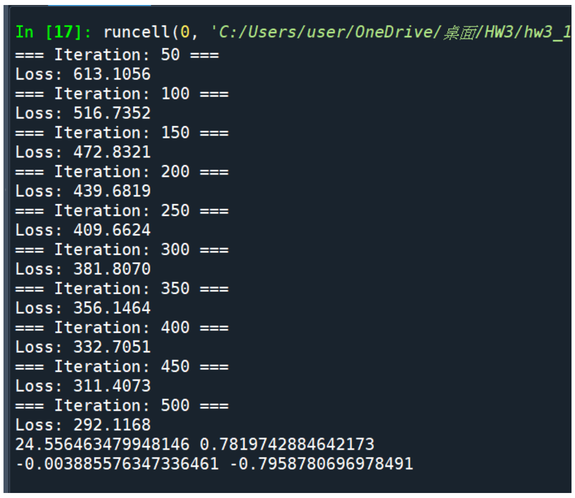
```
In [13]: runcell(0, 'C:/Users/user/OneDrive/桌面/H
=== Iteration: 50 ===
Loss: 604.0665
=== Iteration: 100 ===
Loss: 469.2994
=== Iteration: 150 ===
Loss: 399.8331
=== Iteration: 200 ===
Loss: 347.8617
=== Iteration: 250 ===
Loss: 304.4107
=== Iteration: 300 ===
Loss: 267.8403
=== Iteration: 350 ===
Loss: 237.4926
=== Iteration: 400 ===
Loss: 212.6813
=== Iteration: 450 ===
Loss: 192.6643
=== Iteration: 500 ===
Loss: 176.7208
20.71611432432166 0.409710938905374
-0.002284687474714132 2.2377646075364295
```



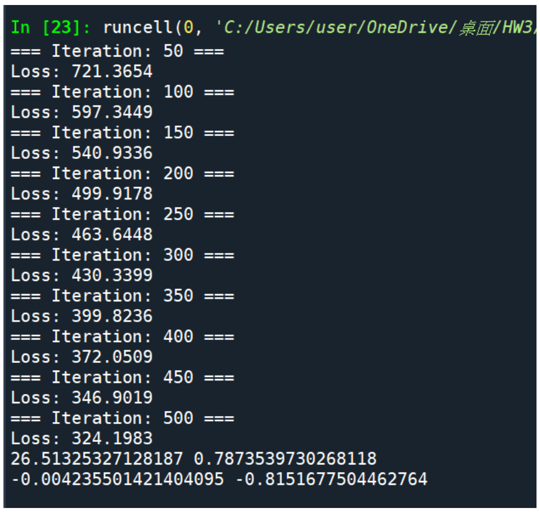
```
In [18]: runcell(0, 'C:/Users/user/OneDrive/桌面
=== Iteration: 50 ===
Loss: 635.4327
=== Iteration: 100 ===
Loss: 537.5749
=== Iteration: 150 ===
Loss: 491.6141
=== Iteration: 200 ===
Loss: 457.2187
=== Iteration: 250 ===
Loss: 426.4977
=== Iteration: 300 ===
Loss: 398.2702
=== Iteration: 350 ===
Loss: 372.4695
=== Iteration: 400 ===
Loss: 349.0742
=== Iteration: 450 ===
Loss: 327.9803
=== Iteration: 500 ===
Loss: 309.0296
25.227535305021373 0.7554319153131697
-0.0037201030397409542 -0.6645947257846434
```



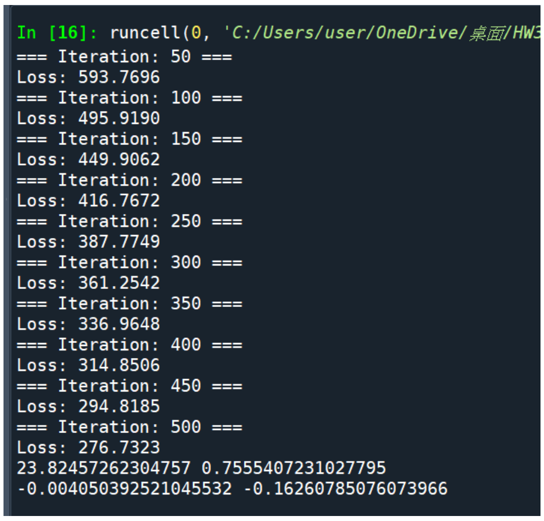
```
In [24]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW
=== Iteration: 50 ===
Loss: 649.7657
=== Iteration: 100 ===
Loss: 545.4641
=== Iteration: 150 ===
Loss: 496.2290
=== Iteration: 200 ===
Loss: 460.4027
=== Iteration: 250 ===
Loss: 428.6726
=== Iteration: 300 ===
Loss: 399.3480
=== Iteration: 350 ===
Loss: 372.2731
=== Iteration: 400 ===
Loss: 347.4618
=== Iteration: 450 ===
Loss: 324.8624
=== Iteration: 500 ===
Loss: 304.3578
25.204292262055827 0.7844375348004528
-0.004029176369649728 -0.5518714222494376
```



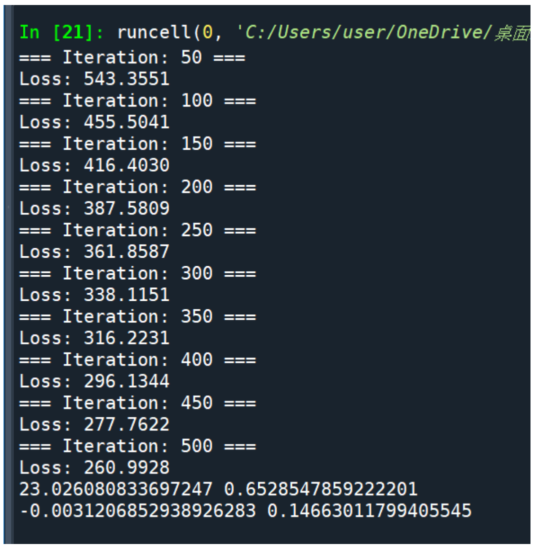
```
In [17]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW3/hw3_1
=== Iteration: 50 ===
Loss: 613.1056
=== Iteration: 100 ===
Loss: 516.7352
=== Iteration: 150 ===
Loss: 472.8321
=== Iteration: 200 ===
Loss: 439.6819
=== Iteration: 250 ===
Loss: 409.6624
=== Iteration: 300 ===
Loss: 381.8070
=== Iteration: 350 ===
Loss: 356.1464
=== Iteration: 400 ===
Loss: 332.7051
=== Iteration: 450 ===
Loss: 311.4073
=== Iteration: 500 ===
Loss: 292.1168
24.556463479948146 0.7819742884642173
-0.003885576347336461 -0.7958780696978491
```



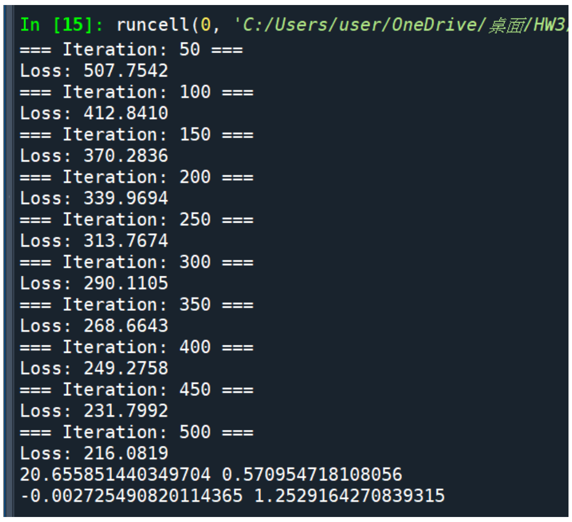
```
In [23]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW3/
=== Iteration: 50 ===
Loss: 721.3654
=== Iteration: 100 ===
Loss: 597.3449
=== Iteration: 150 ===
Loss: 540.9336
=== Iteration: 200 ===
Loss: 499.9178
=== Iteration: 250 ===
Loss: 463.6448
=== Iteration: 300 ===
Loss: 430.3399
=== Iteration: 350 ===
Loss: 399.8236
=== Iteration: 400 ===
Loss: 372.0509
=== Iteration: 450 ===
Loss: 346.9019
=== Iteration: 500 ===
Loss: 324.1983
26.51325327128187 0.7873539730268118
-0.004235501421404095 -0.8151677504462764
```



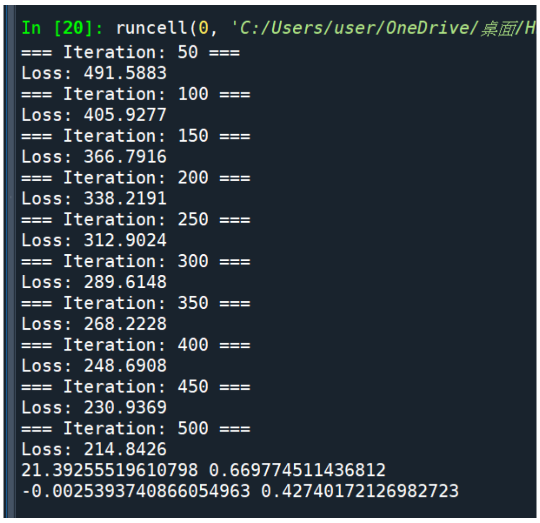
```
In [16]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW3
=== Iteration: 50 ===
Loss: 593.7696
=== Iteration: 100 ===
Loss: 495.9190
=== Iteration: 150 ===
Loss: 449.9062
=== Iteration: 200 ===
Loss: 416.7672
=== Iteration: 250 ===
Loss: 387.7749
=== Iteration: 300 ===
Loss: 361.2542
=== Iteration: 350 ===
Loss: 336.9648
=== Iteration: 400 ===
Loss: 314.8506
=== Iteration: 450 ===
Loss: 294.8185
=== Iteration: 500 ===
Loss: 276.7323
23.82457262304757 0.7555407231027795
-0.004050392521045532 -0.16260785076073966
```



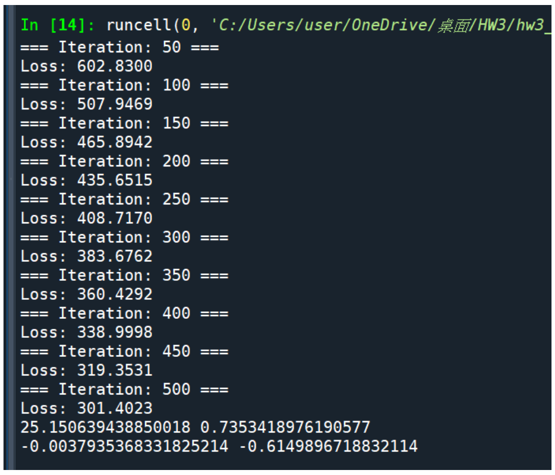
```
In [21]: runcell(0, 'C:/Users/user/OneDrive/桌面
=== Iteration: 50 ===
Loss: 543.3551
=== Iteration: 100 ===
Loss: 455.5041
=== Iteration: 150 ===
Loss: 416.4030
=== Iteration: 200 ===
Loss: 387.5809
=== Iteration: 250 ===
Loss: 361.8587
=== Iteration: 300 ===
Loss: 338.1151
=== Iteration: 350 ===
Loss: 316.2231
=== Iteration: 400 ===
Loss: 296.1344
=== Iteration: 450 ===
Loss: 277.7622
=== Iteration: 500 ===
Loss: 260.9928
23.026080833697247 0.6528547859222201
-0.0031206852938926283 0.14663011799405545
```



```
In [15]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW3/
=== Iteration: 50 ===
Loss: 507.7542
=== Iteration: 100 ===
Loss: 412.8410
=== Iteration: 150 ===
Loss: 370.2836
=== Iteration: 200 ===
Loss: 339.9694
=== Iteration: 250 ===
Loss: 313.7674
=== Iteration: 300 ===
Loss: 290.1105
=== Iteration: 350 ===
Loss: 268.6643
=== Iteration: 400 ===
Loss: 249.2758
=== Iteration: 450 ===
Loss: 231.7992
=== Iteration: 500 ===
Loss: 216.0819
20.655851440349704 0.570954718108056
-0.002725490820114365 1.2529164270839315
```



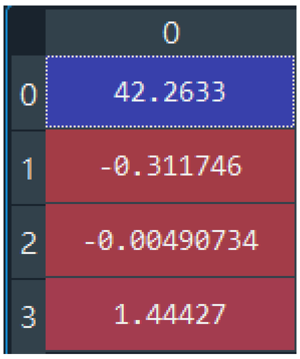
```
In [20]: runcell(0, 'C:/Users/user/OneDrive/桌面/H
=== Iteration: 50 ===
Loss: 491.5883
=== Iteration: 100 ===
Loss: 405.9277
=== Iteration: 150 ===
Loss: 366.7916
=== Iteration: 200 ===
Loss: 338.2191
=== Iteration: 250 ===
Loss: 312.9024
=== Iteration: 300 ===
Loss: 289.6148
=== Iteration: 350 ===
Loss: 268.2228
=== Iteration: 400 ===
Loss: 248.6908
=== Iteration: 450 ===
Loss: 230.9369
=== Iteration: 500 ===
Loss: 214.8426
21.39255519610798 0.669774511436812
-0.002539374086605963 0.42740172126982723
```



```
In [14]: runcell(0, 'C:/Users/user/OneDrive/桌面/HW3/hw3_
=== Iteration: 50 ===
Loss: 602.8300
=== Iteration: 100 ===
Loss: 507.9469
=== Iteration: 150 ===
Loss: 465.8942
=== Iteration: 200 ===
Loss: 435.6515
=== Iteration: 250 ===
Loss: 408.7170
=== Iteration: 300 ===
Loss: 383.6762
=== Iteration: 350 ===
Loss: 360.4292
=== Iteration: 400 ===
Loss: 338.9998
=== Iteration: 450 ===
Loss: 319.3531
=== Iteration: 500 ===
Loss: 301.4023
25.150639438850018 0.7353418976190577
-0.0037935368331825214 -0.6149896718832114
```

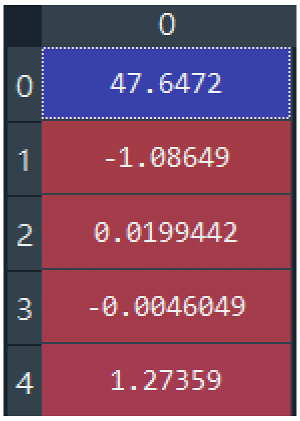
The bottom two lines in every picture indicates the values of B0，B1，B2，B3
For the ten iterations

5.



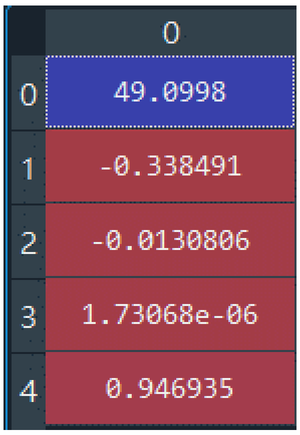
	0
0	42.2633
1	-0.311746
2	-0.00490734
3	1.44427

B0:42.2633, B1:-0.3117, B2:-0.004, B3:1.444



	0
0	47.6472
1	-1.08649
2	0.0199442
3	-0.0046049
4	1.27359

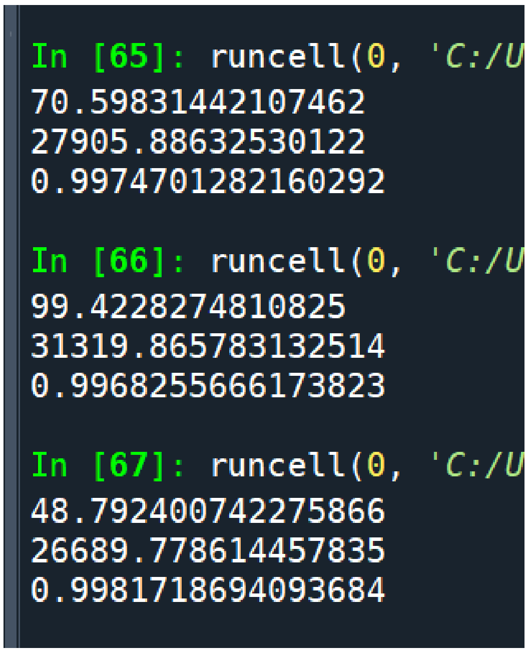
B4:47.6472, B5:1.086, B6:-0.019, B7:-0.004, B8:1.2735



	0
0	49.0998
1	-0.338491
2	-0.0130806
3	1.73068e-06
4	0.946935

B9:49.0998, B10:-0.3384, B11:-0.013, B12:0.00000173,

B13:0.9469



```
In [65]: runcell(0, 'C:/U
70.59831442107462
27905.88632530122
0.9974701282160292

In [66]: runcell(0, 'C:/U
99.4228274810825
31319.865783132514
0.9968255666173823

In [67]: runcell(0, 'C:/U
48.792400742275866
26689.778614457835
0.9981718694093684
```

65 -> the first model

66 -> the second model

67 -> the third model

The first number represents the numerator of Erse, the second number represents The denominator of Erse, and the third value is R-square

REPORT:

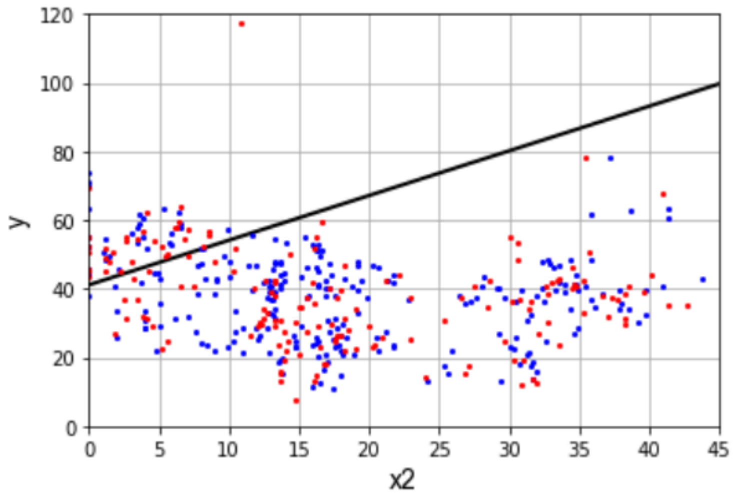
1.

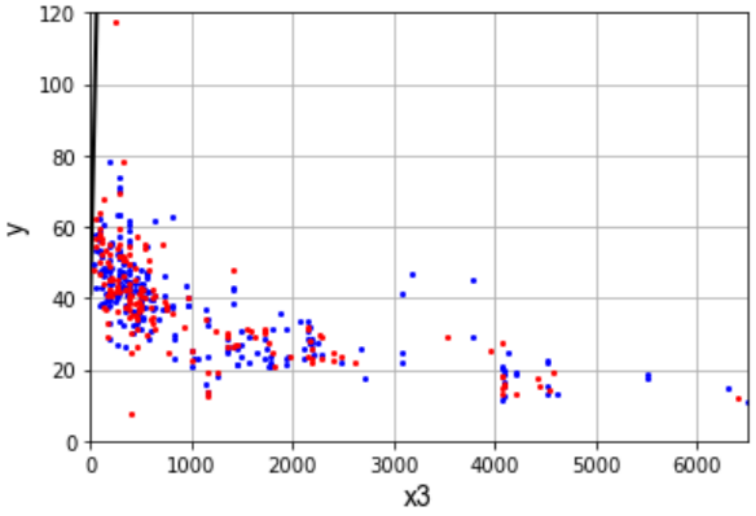
The two set of numbers are different because they use different methods and They have different losses .

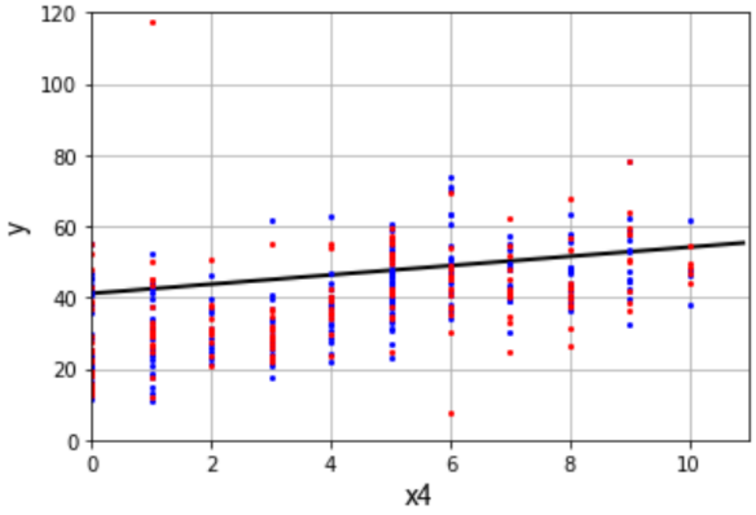
2.

I think the least square method is better because it does not have to iterate through for a lot of times , it puts data in a matrix to produce parameters , unlike gradient method , if the times of iteration is not big enough , like about 500 , it only converges to the local min instead of the global min . Therefore we get an inaccurate answer compared to least square method . Convergence accuracy of forecast model is assumed to be 0.001 , which is the learning rate , we have to shift the data to near 10 to get a better result , otherwise the loss would be larger after every iteration.

The least square method gets an average loss of 70 , which is much more stable than gradient method , and from the plots below , we can infer that the housing prices is more relevant to data x4 compared to x2 and x3 .


y
x2
0
5
10
15
20
25
30
35
40
45
0
20
40
60
80
100
120


y
x3
0
1000
2000
3000
4000
5000
6000
0
20
40
60
80
100
120


y
x4
0
2
4
6
8
10
0
20
40
60
80
100
120