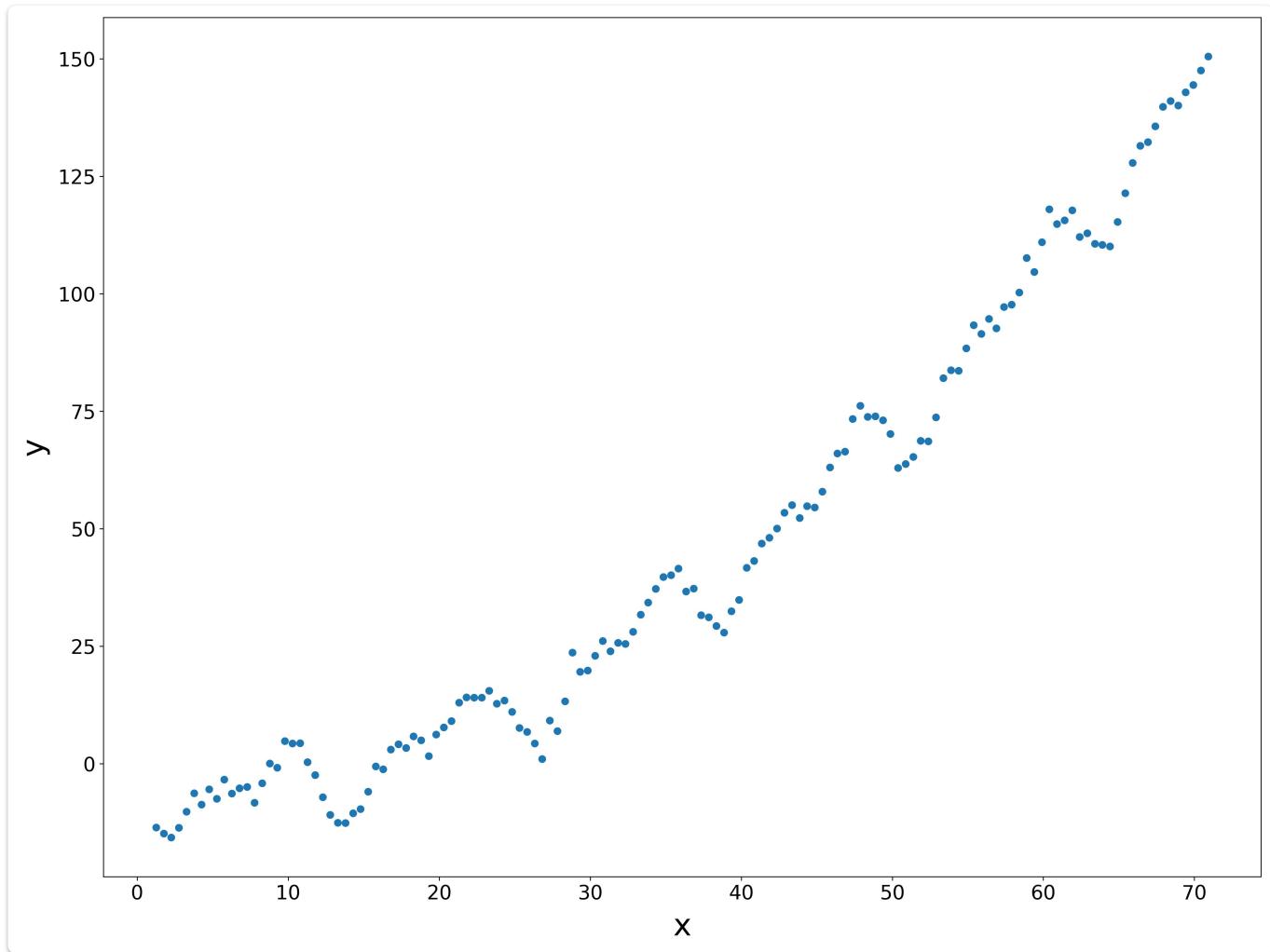


중앙대학교 예술공학대학 2023-2 인공지능과 머신러닝 과제# 1

목표

- 주어진 데이터를 잘 나타내는 한 편, 예측력도 높은 함수 $f(x)$ 를 설계하는 과제입니다.
- 모든 코드는 `regression.py` 파일에 작성하며, 파일명을 변경하면 안 됩니다. 파일명을 변경하는 경우 평가 시스템에서 오류가 발생하게 됩니다.
- 과제의 최종 점수는 제출 기한 이후 최종 순위에 따라 차등 결정됩니다.
- 제출 기한은 **2023년 10월 24일 화요일 오후 11시 59분**입니다.



문제(1)

- 본 문제에서는 제공된 데이터(훈련 데이터)를 `func` 함수가 잘 **설명**할 수 있는지 테스트합니다.
- 본 문제의 평가 데이터는 훈련 데이터를 포함하며, 훈련 데이터에는 있지만 훈련 데이터의 x 및 y 의 범위에 존재하는 **별도의 데이터**도 포함합니다.

- 평가 데이터의 범위는 아래와 같습니다.

$$\begin{aligned} 1 < x < 72 \\ -19 < y < 157 \end{aligned}$$

문제(2)

- 본 문제에서는 `func` 함수가 훈련 데이터의 범위를 벗어나는 데이터를 잘 예측할 수 있는지 테스트 합니다.
- 본 문제의 평가 데이터는 훈련 데이터에는 없는 데이터로서, x 와 y 의 범위는 다음과 같습니다.

$$\begin{aligned} 71 < x < 101 \\ 156 < y < 304 \end{aligned}$$

접근

사전 접근

다들 과제를 어떻게 하나

최적의 함수를 찾기 위해 다른 사람들이 다양한 방법으로 접근하더라

- 2차 이하의 함수
 - 가장 기본적인 접근 방법
 - 이걸로 고득점은 당연히 안된다
- 3차 이상의 함수
 - 다들 인터넷에서 모델 피팅을 검색해서 함수를 찾더라
 - 데이터의 모양이 꾸불꾸불한데 이거때문에 계수를 무진장 늘려놓으면 오버피팅이 될 거 같다

아이디어

근데 함수를 굳이 1개의 다항식으로 표현해야 할까?

구간함수를 사용해도 되겠는데?

1번째 접근

아이디어

- 구간함수를 통해 모델을 피팅한다
- 데이터를 직접 선별해 눈에 보이는 패턴대로 구간을 정의한다
- 일단은 각 구간 별 함수를 간단하게 근사해서 동향을 파악해본다

구간 나누기

데이터를 눈에 보이는 대로 3가지 패턴으로 구분했다

```

for i in x:
    if i < 2.256:      #      3
        y.append(cul_function(i, 0))
    elif i < 7.770:    # +5.514  1
        y.append(cul_function(i, 1))
    elif i < 10.276:   # +2.506  2
        y.append(cul_function(i, 2))
    elif i < 13.784:   # +3.508  3
        y.append(cul_function(i, 3))
    elif i < 19.298:   # +5.514  1
        y.append(cul_function(i, 4))
    elif i < 22.306:   # +3.008  2
        y.append(cul_function(i, 5))
    elif i < 26.817:   # +4.511  3
        y.append(cul_function(i, 6))
    elif i < 31.830:   # +5.013  1
        y.append(cul_function(i, 7))
    elif i < 35.839:   # +4.009  2
        y.append(cul_function(i, 8))
    elif i < 38.847:   # +3.008  3
        y.append(cul_function(i, 9))
    elif i < 44.862:   # +6.015  1
        y.append(cul_function(i, 10))
    elif i < 48.370:   # +3.508  2
        y.append(cul_function(i, 11))
    elif i < 50.877:   # +2.507  3
        y.append(cul_function(i, 12))
    elif i < 57.394:   # +6.517  1
        y.append(cul_function(i, 13))
    elif i < 60.902:   # +3.508  2
        y.append(cul_function(i, 14))
    elif i < 64.411:   # +3.509  3
        y.append(cul_function(i, 15))
    elif i < 69.423:   # +5.012  1
        y.append(cul_function(i, 16))

```

- 1번 패턴
 - y값이 가파르게 증가했다가 증가폭이 서서히 감소하는 형태
 - 2차 함수로 근사되는 형태
 - 임의의 3개 점을 잡아 2차 함수를
- 2번 패턴
 - y값이 일정한 속도로 증가하는 형태
 - 1차 함수로 근사되는 형태
- 3번 패턴
 - y값이 일정한 속도로 감소하는 형태
 - 1차 함수로 근사되는 형태

위 과정을 통해 구간 2차 함수로 주어진 데이터를 충분히 근사할 수 있다고 판단했기 때문에 각 구간별로 제곱 오차를 최소화하는 2차식의 계수를 구한다
다항식은 numpy에서 제공하는 [polyfit](#)을 통해 계산했다

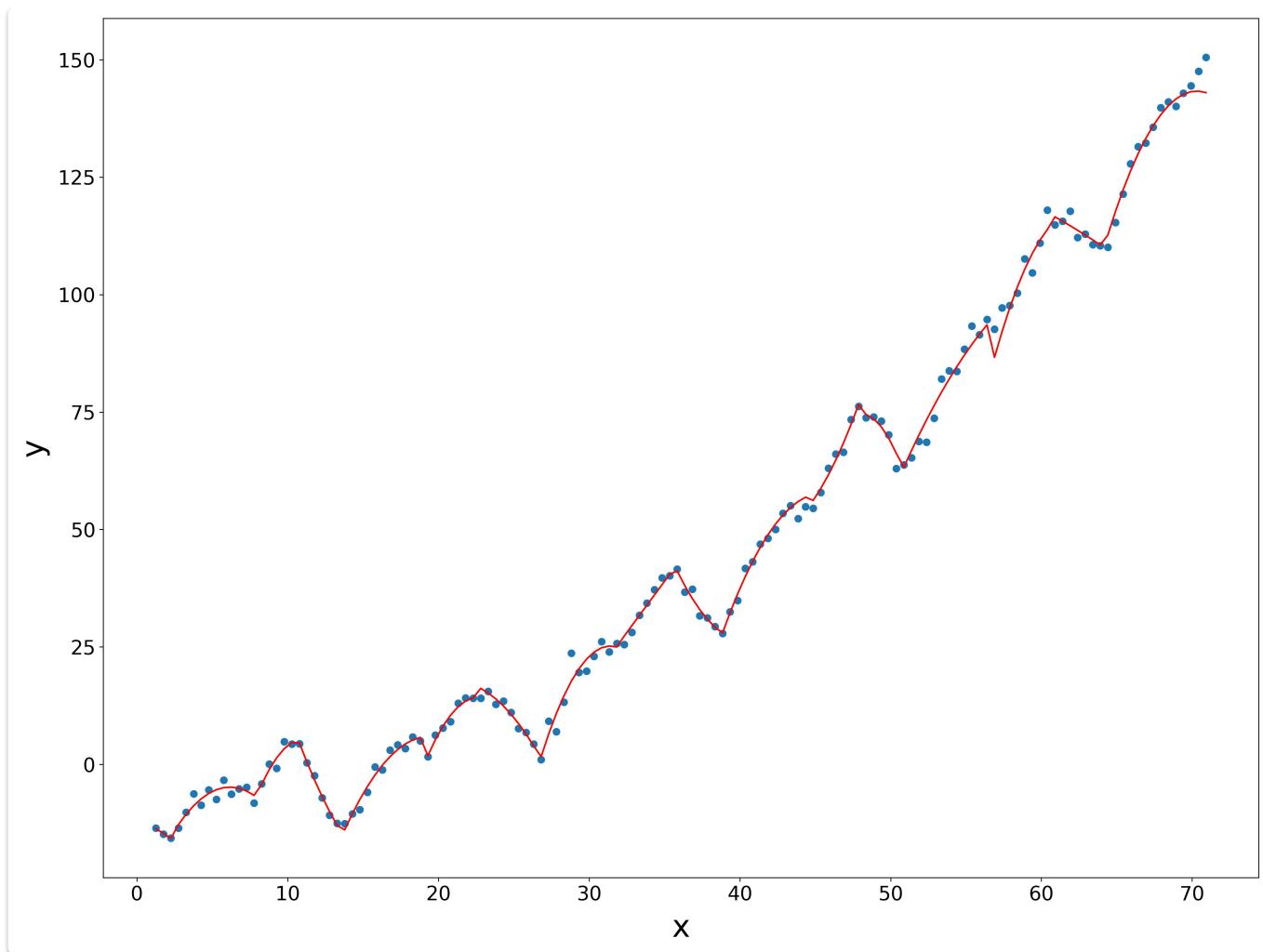
```
def make_linear_function(s_idx, e_idx, deg=5):
    x_hat = x[s_idx : e_idx]
    y_hat = y[s_idx : e_idx]
    return np.polyfit(x_hat, y_hat, deg)

print(make_linear_function(0,3,1))      # 3
print(make_linear_function(2,13,2))     # 1
print(make_linear_function(13,19,2))    # 2
print(make_linear_function(19,25,2))    # 3
print(make_linear_function(25,36,2))    # 1
print(make_linear_function(36,44,2))    # 2
print(make_linear_function(44,52,2))    # 3
print(make_linear_function(51,60,2))    # 1
print(make_linear_function(60,70,2))    # 2
print(make_linear_function(69,76,2))    # 3
print(make_linear_function(75,85,2))    # 1
print(make_linear_function(85,94,2))    # 2
print(make_linear_function(94,100,2))   # 3
print(make_linear_function(98,115,2))   # 1
print(make_linear_function(115,122,2))  # 2
print(make_linear_function(118,127,2))  # 3
print(make_linear_function(125,136,2))  # 1
```

구간별로 생성한 2차식의 계수는 다음과 같다

```
poly_array = np.array([
    [ 0.0,      -2.11292974,      -11.00121303],
    [-0.68538924,      8.45635734,      -30.92506365],
    [-1.21803016,      27.05919758,      -144.77265501],
    [ 0.55494366,      -20.36227483,      159.54835887],
    [-0.62911887,      24.42150183,      -231.034005],
    [-1.08832353,      49.37887566,      -545.774095],
    [-0.49595229,      20.88081602,      -202.07838248],
    [-1.11963727,      70.32438121,      -1079.07350409],
    [-8.89130482e-03,  4.95667242e+00,      -1.23628476e+02],
    [ 7.26066341e-01,  -5.86331035e+01,      1.20996091e+03],
    [-6.79371960e-01,  6.17741759e+01,      -1.34656157e+03],
    [ 6.91600166e-01,  -5.73542382e+01,      1.23729090e+03],
    [-1.55646517e+00,  1.49570247e+02,      -3.51864591e+03],
    [-2.97719384e-01,  3.74306195e+01,      -1.07046086e+03],
    [-1.06630822e+00,  1.32826101e+02,      -4.01877629e+03],
    [-0.04905084,      4.13564291,      46.62817084],
    [-8.80492780e-01,  1.23816129e+02,      -4.20946383e+03]
])
```

구간별 2차식을 적용한 함수는 아래와 같다



오버피팅을 최소화하면서 주어진 데이터를 피팅했다

구간 예측하기

각 패턴이 어느 정도의 주기로 찾아오는지 파악한다

1->2	2->3	3->1	1->1
-	-	5.514	5.514
2.506	3.508	5.514	11.528
3.008	4.511	5.013	12.532
4.009	3.008	6.015	13.032
3.508	2.507	6.517	12.030
3.508	3.509	5.012	12.531

- 1->2에서, 뚜렷한 패턴은 보이지 않으나 뒤로 갈수록 증가폭이 약 3.5에 가까워지는 것으로 보인다. 일단은 약 3.5의 증가폭을 가지는 것으로 예상해본다.
- 2->3에서, 역시 뚜렷한 패턴은 보이지 않는다. 억지로 규칙을 끼워맞춘다면 +1.0, -1.5, -0.5, +1.0으로 증가폭이 변화하는데 약 2.5에서 3.5 사이의 증가폭으로 예상하고 진행하는 것이 좋아보인다.

- 3->1에서, 마찬가지로 뚜렷한 패턴이 보이지 않는다. 약 5.5에서 6.5의 증가폭을 보일 것으로 보인다.
- 전체적으로, 하나의 사이클이 약 11.5에서 13.0 주기로 진행된다. 역시 뚜렷한 패턴은 없다

패턴이 크게 존재하지 않아서 임의로 계산해본다

- 1->2 약 3.0~4.0
- 2->3 약 2.5~3.5
- 3->1 약 5.5~6.5
- 전체 사이클 약 12.0~13.0

위에서 $x=69$ 까지의 데이터를 피팅했고, 패턴이 1번으로 끝났으며, 문제(2)의 데이터의 x 가 101까지이므로 남은 패턴은 2-3-1-2-3-1-2-3이다.

각 구간 별 구간함수를 추정하기 위해 각 구간 별 이차항의 계수와 꼭짓점의 좌표를 정리한다

p	a	x	y
1	-0.68538924	6.1690181625845195	-4.8413526401175435
2	-1.21803016	11.107769934038414	5.511015649164503
3	0.55494366	18.346254131455435	-27.2373754928593
1	-0.62911887	19.409290512935975	5.968006890333752
2	-1.08832353	22.68575212188971	14.324371640186527
3	-0.49595229	21.05123460565128	17.705095917230867
1	-1.11963727	31.404984048985796	25.19453098742326
2	-0.00889130482	278.73706505093145	567.175685384849
3	0.726066341	40.377235652630254	26.239591467720135
1	-0.67937196	45.4641783420087	57.69450502410782
2	0.691600166	41.46488174787396	48.19754764880228
3	-1.55646517	48.048054618530266	74.63378858153071
1	-0.297719384	62.86224799524643	106.02558281235348
2	-1.06630822	62.28316471198168	117.6386733166567
3	-0.04905084	42.15669813197898	133.80069570926455
1	-0.88049278	70.31070090092051	143.33557641439572

이를 패턴별로 정리한다

p	a	x	y	x 증가량	y 증가량
1	-0.68538924	6.1690181625845195	-4.8413526401175435	-	-
1	-0.62911887	19.409290512935975	5.968006890333752	13.240	10.809

p	a	x	y	x 증가량	y 증가량
1	-1.11963727	31.404984048985796	25.19453098742326	11.995	19.226
1	-0.67937196	45.4641783420087	57.69450502410782	14.060	32.500
1	-0.297719384	62.86224799524643	106.02558281235348	17.398	48.331
1	-0.88049278	70.31070090092051	143.33557641439572	7.448	37.31

p	a	x	y	x 증가량	y 증가량
2	-1.21803016	11.107769934038414	5.511015649164503	-	-
2	-1.08832353	22.68575212188971	14.324371640186527	11.578	8.813
2	-0.00889130482	278.73706505093145	567.175685384849	256.052	552.851
2	0.691600166	41.46488174787396	48.19754764880228	-237.273	-518.978
2	-1.06630822	62.28316471198168	117.6386733166567	20.819	69.441

p	a	x	y	x 증가량	y 증가량
3	0.55494366	18.346254131455435	-27.2373754928593	-	-
3	-0.49595229	21.05123460565128	17.705095917230867	2.705	44.942
3	0.726066341	40.377235652630254	26.239591467720135	19.326	8.534
3	-1.55646517	48.048054618530266	74.63378858153071	7.671	48.394
3	-0.04905084	42.15669813197898	133.80069570926455	-5.892	59.167

패턴이 전혀 짐작가지 않기 때문에 임의로 함수를 추정해본다.

- 1번 패턴
 - 이차항의 계수를 -0.65로 추정한다
 - 꼭짓점의 x값이 약 13만큼 이동한다고 추정한다
 - 꼭짓점의 y값이 약 50, 60만큼 이동한다고 추정한다
- 2번 패턴
 - 이차항의 계수를 -1로 추정한다
 - 꼭짓점의 x값이 약 10만큼 이동한다고 추정한다
 - 꼭짓점의 y값이 약 40만큼 이동한다고 추정한다
- 3번 패턴
 - 이차항의 계수를 -0.05로 추정한다
 - 꼭짓점의 x값이 약 40에 있다고 추정한다
 - 꼭짓점의 y값이 약 60만큼 이동한다고 추정한다

추정한 구간함수의 이차항의 계수와 꼭짓점은 다음과 같다

p	a	x	y
2	-1	75	160
3	-0.5	40	190

p	a	x	y
1	-0.65	83	190
2	-1	90	200
3	-0.5	40	250
1	-0.65	96	250
2	-1	105	240
3	-0.5	40	310

이 값을 이차식으로 변형한다

```
def transformation(a, x, y):
    b = -2.0 * a * x
    c = y + (b ** 2) / (4.0 * a)
    return [a, b, c]

l = []

l.append(transformation(-1, 80, 160))
l.append(transformation(-0.5, 40, 190))
l.append(transformation(-0.65, 83, 190))
l.append(transformation(-1, 100, 200))
l.append(transformation(-0.5, 40, 250))
l.append(transformation(-0.65, 96, 250))
l.append(transformation(-1, 120, 240))
l.append(transformation(-0.5, 40, 310))
```

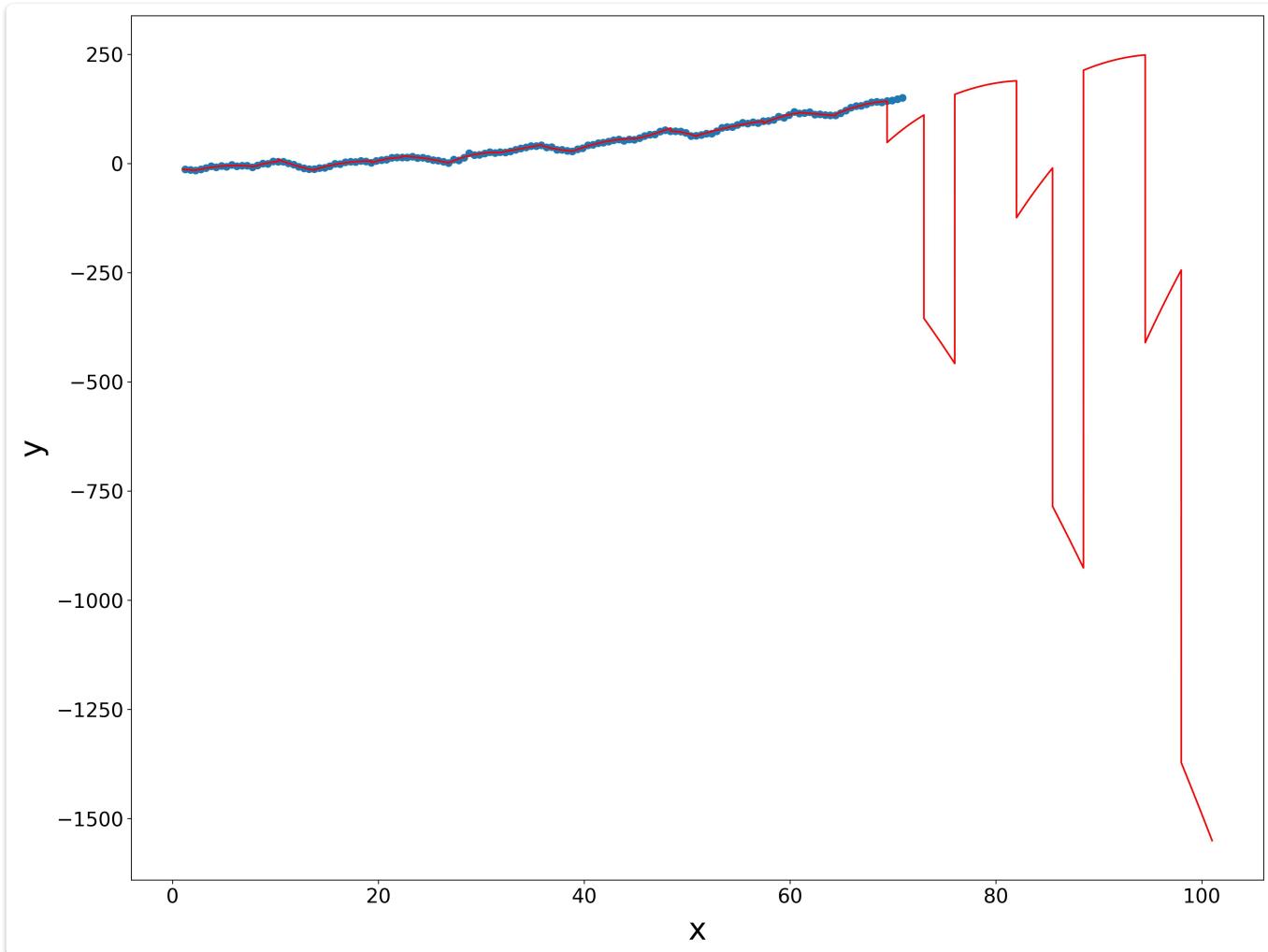
이렇게 생성한 이차식의 계수들을 배열에 넣는다

```
[-1, 160.0, -6240.0],
[-0.5, 40.0, -610.0],
[-0.65, 107.9, -4287.85],
[-1, 200.0, -9800.0],
[-0.5, 40.0, -550.0],
[-0.65, 124.8000000000001, -5740.400000000001],
[-1, 240.0, -14160.0],
[-0.5, 40.0, -490.0]
```

구간은 위에서 예측한 대로 작성한다

```
elif i < 73.0:
    y.append(cul_function(i, 17))
elif i < 76.0:
    y.append(cul_function(i, 18))
elif i < 82.0:
    y.append(cul_function(i, 19))
elif i < 85.5:
    y.append(cul_function(i, 20))
elif i < 88.5:
    y.append(cul_function(i, 21))
elif i < 94.5:
    y.append(cul_function(i, 22))
elif i < 98.0:
    y.append(cul_function(i, 23))
elif i < 101.0:
    y.append(cul_function(i, 24))
```

이 상태로 그래프를 한번 뽑아봤다



망했다

결국 직감의 힘을 빌려 수치를 세부 조정해 노가다해서 그럴듯한 함수 개형을 찾았다

```

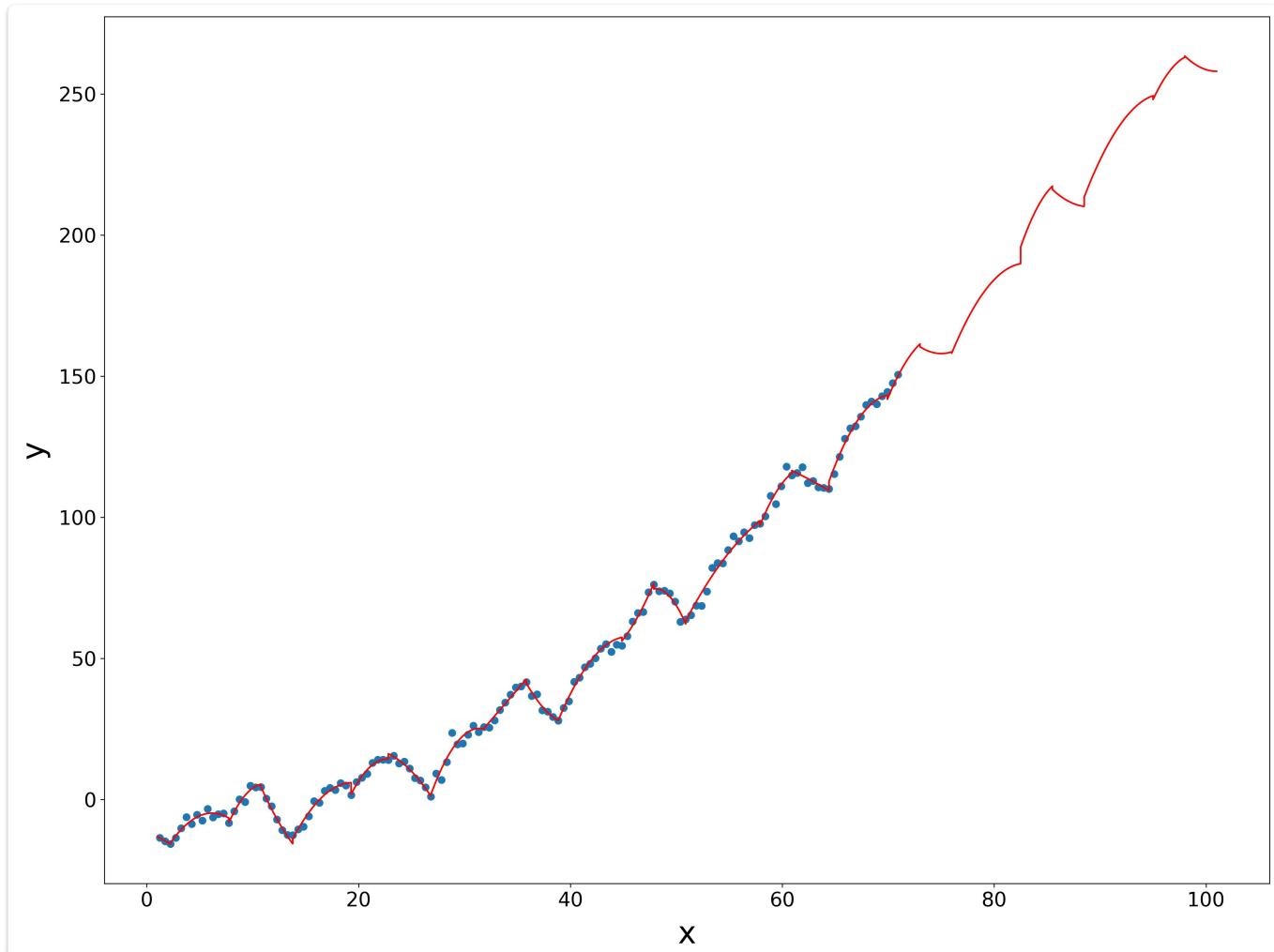
l.append(transformation(-0.9, 75.0, 165.0))      # 2
l.append(transformation(0.6, 75.0, 158.0))      # 3
l.append(transformation(-0.65, 83.0, 190.0))    # 1
l.append(transformation(-1.2, 87.0, 220.0))      # 2
l.append(transformation(0.5, 89.0, 210.0))      # 3
l.append(transformation(-0.65, 96.0, 250.0))    # 1
l.append(transformation(-1.0, 99.0, 264.0))      # 2
l.append(transformation(0.6, 101.0, 258.0))      # 3

```

```

[-0.9, 135.0, -4897.5],
[0.6, -90.0, 3533.0],
[-0.65, 107.9, -4287.85],
[-1.2, 208.7999999999998, -8862.8],
[0.5, -89.0, 4170.5],
[-0.65, 124.8000000000001, -5740.400000000001],
[-1.0, 198.0, -9537.0],
[0.6, -121.1999999999999, 6378.599999999985]

```



이정도면 그럴듯하게 생겼는데 과연?

결과 및 피드백

[2023-10-21 01:23:13]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.520024

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 328.292198

- Metric: 331.81222200

- Rank: 47

- Score: 72.5

2번 문제에서 굉장히 낮은 점수를 받았다.

그래프의 개형이 완전히 잘못된 것으로 보인다

2번째 접근

문제점

앞서 제작한 함수는 다음 문제가 있었다

- 1번 문제는 그럴듯하게 설명됐다
- 2번 문제가 제대로 설명되지 않았다

이후 구간 예측 실패에 대한 가설은 다음과 같다

- 증가량이 너무 작다
- 제기된 문제의 구간은 다음과 같다

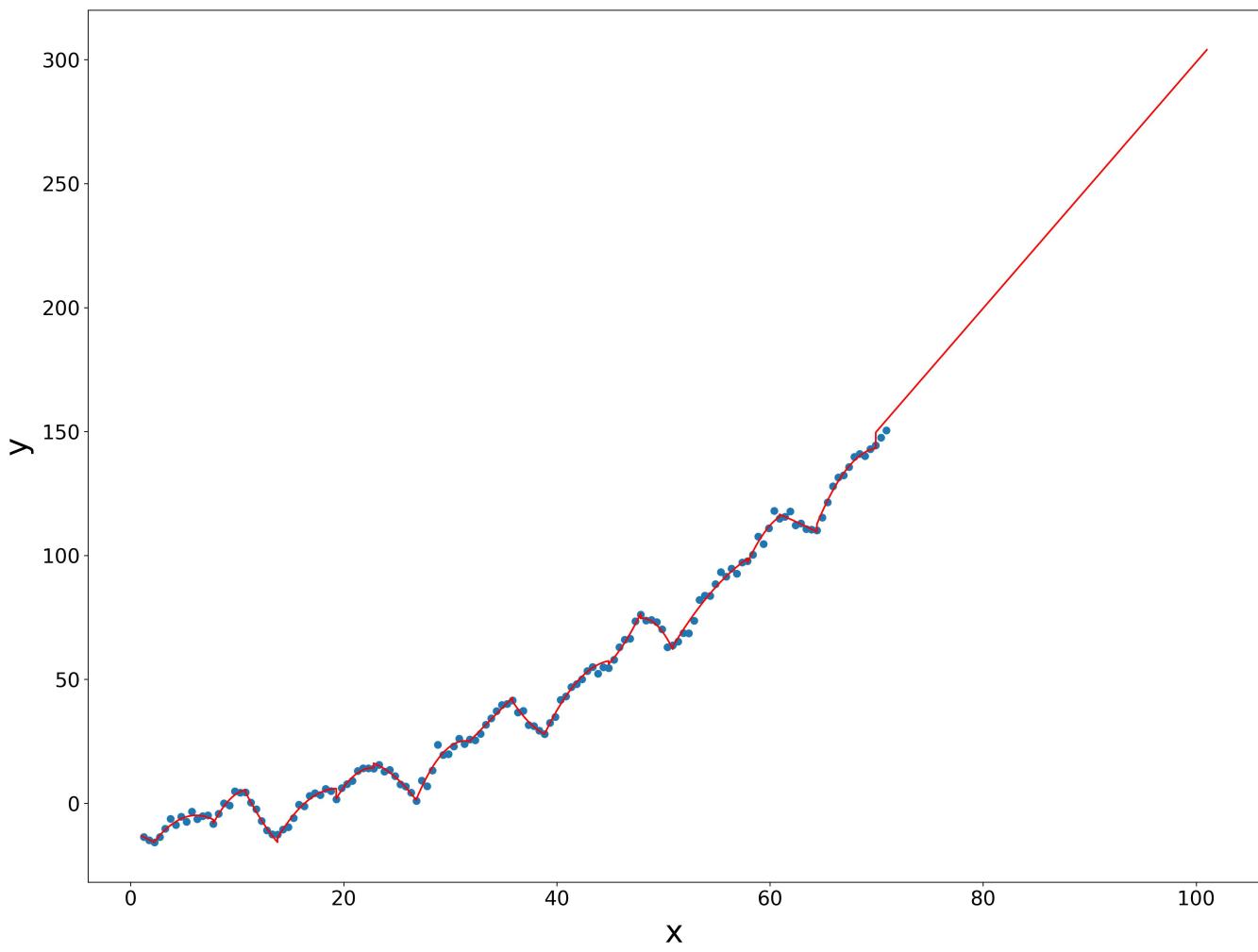
$$\begin{aligned} 71 < x < 101 \\ 156 < y < 304 \end{aligned}$$

- 단순히 생각해본다면 x가 101일 때 y가 304까지 갈 수 있다는 뜻이다
- 근데 지금 함수는 y가 260까지밖에 증가하지 않는다
- 좀 더 상승량이 많아야 될 것 같다

이후 구간이 어떻게 생겼을까?

이후 구간이 어떻게 생겼는지 확인하기 위해 일단 확인 작업이 필요하다

예측 범위인 (70, 150)에서 (101, 304)까지 직선을 그려서 점수를 확인해본다



[2023-10-21 01:44:20]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.659482

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 66.777712

- Metric: 70.43719400
- Rank: 42
- Score: 75.0

확실히 기존 함수보다 점수가 높다

기존 개형에서 y값의 상승량을 더 높여보고 테스트해봐야겠다

아이디어

그래프를 개선하기 위해서 다음 방법을 사용해본다

- 각 패턴 별 이차항의 계수는 그대로일 것으로 간주한다
- 예측 구간에서 각 구간의 경계마다 가질 수 있는 데이터를 예측한다
- 예측한 데이터들을 지나도록 구간함수를 수정한다

가상 데이터 예측하기

각 구간을 결정하는 대표 데이터는 다음과 같다

p	x	y	비고
3->1	2.255	-15.703	
1->2	7.769	-8.266	
2->3	10.275	4.332	o
3->1	13.784	-12.604	
1->2	19.298	1.614	o
2->3	23.308	15.537	o
3->1	26.817	1.029	
1->2	32.330	25.492	
2->3	35.839	41.558	
3->1	38.847	27.899	
1->2	44.862	54.533	
2->3	48.370	73.817	o
3->1	50.375	62.959	
1->2	56.892	92.647	
2->3	61.904	117.778	o
3->1	64.411	110.089	
1->2	68.922	140.070	

이를 구간별로 재분류한다

p	x	y	x 증가량	y 증가량	비고
1->2	7.769	-8.266	-	-	
1->2	19.298	1.614	11.529	9.88	o
1->2	32.330	25.492	13.032	23.878	
1->2	44.862	54.533	12.532	29.041	

p	x	y	x 증가량	y 증가량	비고
1->2	56.892	92.647	12.030	38.114	
1->2	68.922	140.070	12.030	47.423	
p	x	y	x 증가량	y 증가량	비고
2->3	10.275	4.332	-	-	○
2->3	23.308	15.537	13.033	11.205	○
2->3	35.839	41.558	12.531	26.021	
2->3	48.370	73.817	12.531	32.259	○
2->3	61.904	117.778	13.534	43.961	○
p	x	y	x 증가량	y 증가량	비고
3->1	2.255	-15.703	-	-	
3->1	13.784	-12.604	11.529	3.099	
3->1	26.817	1.029	13.033	13.633	
3->1	38.847	27.899	12.030	26.870	
3->1	50.375	62.959	11.528	35.060	
3->1	64.411	110.089	14.036	47.130	

증가량의 변화량에 일관성이 없다

임의로 다음과 같이 예측한다

- x 증가량은 약 12.5에서 약 13.5다
- y 증가량은 약 9.0~10.0씩 상승한다

따라서 예측되는 8개의 점은 다음과 같다

p	x	y
2->3	74	170
3->1	77	167
1->2	81	207
2->3	86.5	233
3->1	89.5	234
1->2	93.5	284
2->3	99	306
3->1	102	311

해당 점을 통해 개선한 구간함수는 다음과 같다

```

def make_function(a, b, p):
    p = np.array([a[1] - p * (a[0] ** 2), b[1] - p * (b[0] ** 2)])
    q = np.array([[a[0], 1], [b[0], 1]])
    n = np.dot(np.linalg.inv(q), p)
    return n

l = []

l.append(make_function([70.927, 150.496], [74, 170], -0.9))      # 2
l.append(make_function([74, 170], [77, 167], 0.6))                  # 3
l.append(make_function([77, 167], [81, 207], -0.65))                # 1
l.append(make_function([81, 207], [86.5, 233], -1.2))                # 2
l.append(make_function([86.5, 233], [89.5, 234], 0.5))                # 3
l.append(make_function([89.5, 234], [93.5, 284], -0.65))                # 1
l.append(make_function([93.5, 284], [99, 306], -1.0))                  # 2
l.append(make_function([99, 306], [102, 311], 0.6))                  # 3

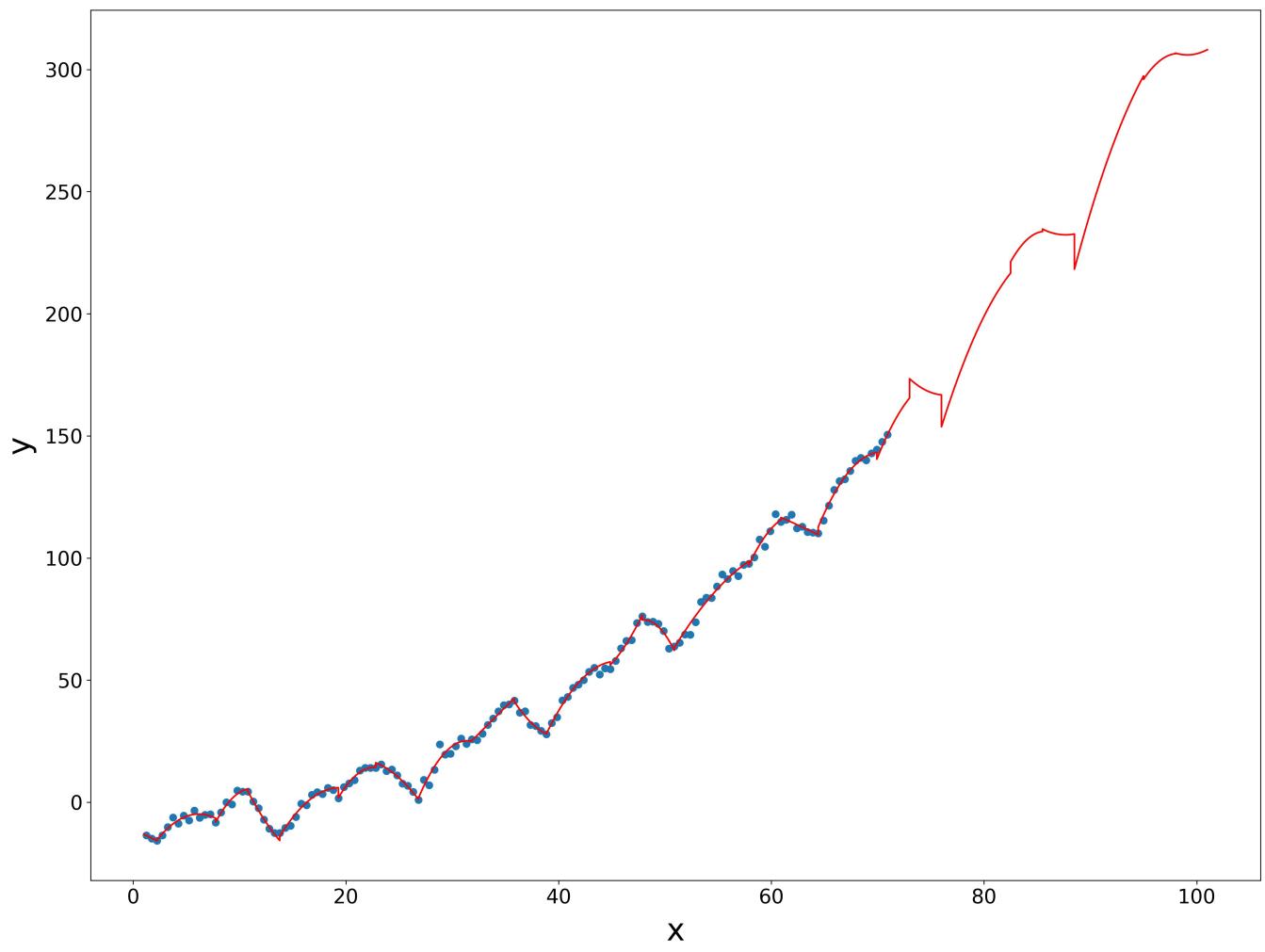
for i in l:
    print(i,end=',\n')

```

```

[-0.9, 136.78119229, -5023.40822929],
[0.6, -91.6, 3662.8],
[-0.65, 112.7, -4657.05],
[-1.2, 205.72727273, -8583.70909091],
[0.5, -87.66666667, 4075.04166667],
[-0.65, 131.45, -6324.1125],
[-1.0, 196.5, -9346.5],
[0.6, -118.93333333, 6199.8]

```



일부 튜는 값이 보이나 일단은 이걸로 점수를 확인해본다

```
[2023-10-21 02:58:56]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 3.523050
```

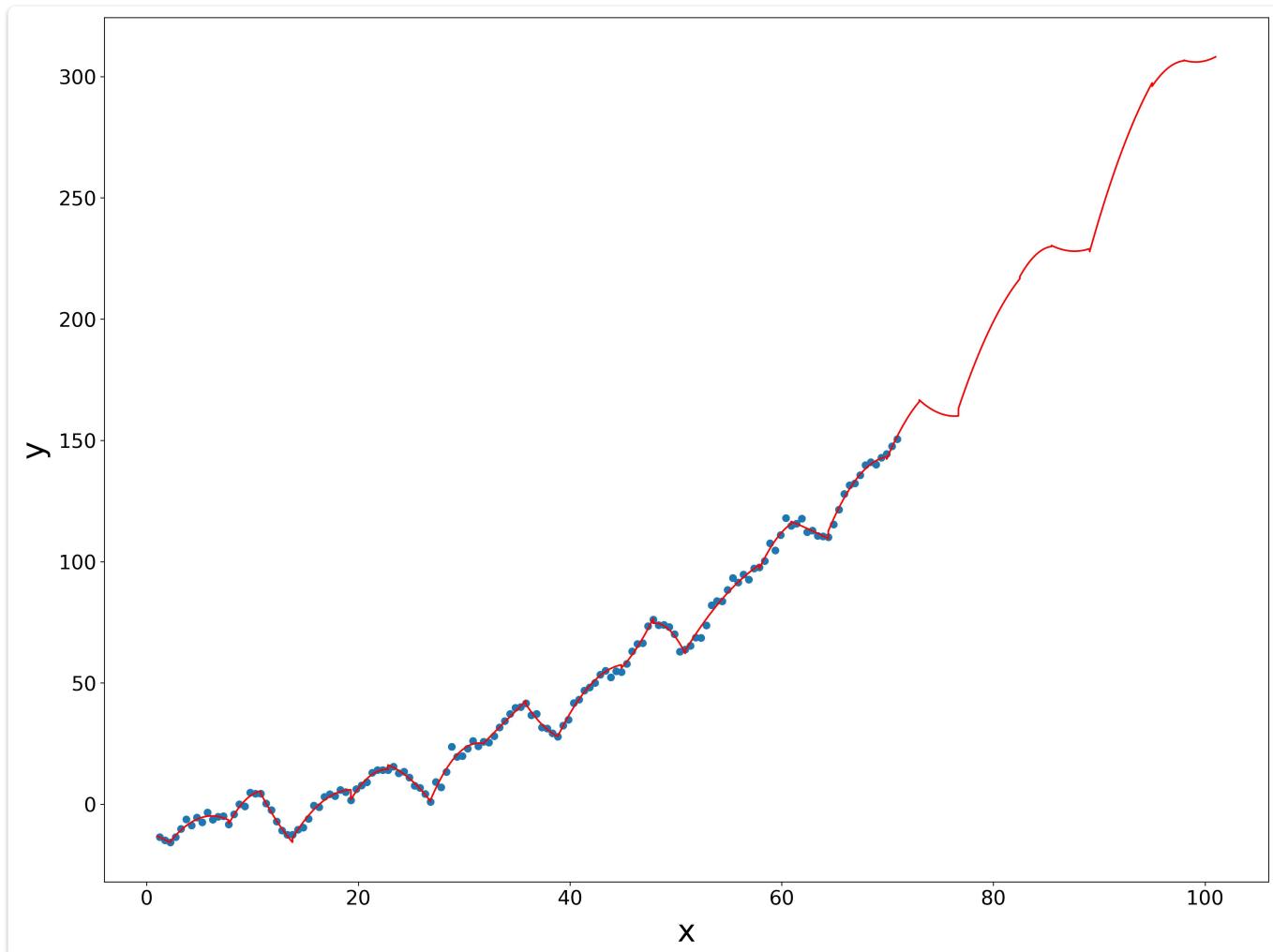
```
[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 110.804901
```

```
- Metric: 114.32795100
- Rank: 44
- Score: 75.0
```

처음 함수보다 개선됐으나 일차 함수보다는 낮은 점수다

튀어나온 값을 일부 정리해본다

```
[-0.85, 129.1822371627778, -4734.687634162317],  
[0.6, -91.6, 3656.066666666666],  
[-0.65, 112.7, -4657.05],  
[-1.2, 205.72727273, -8587.439738525793],  
[0.5, -87.66666667, 4070.722222514444],  
[-0.65, 131.45, -6324.112500000001],  
[-1.0, 196.5, -9346.5],  
[0.6, -118.93333333, 6199.800000000001],
```



```
[2023-10-21 03:14:10]  
[SECURITY] Checking security issues has passed!  
[PROBLEM_01] Testing observed data  
Evaluation metric: 3.452964
```

```
[SECURITY] Checking security issues has passed!  
[PROBLEM_02] Testing unseen data  
Evaluation metric: 103.874445
```

- Metric: 107.32740900
- Rank: 44
- Score: 75.0

튀어나온 값을 아래로 내리니 Metric이 더 감소했다

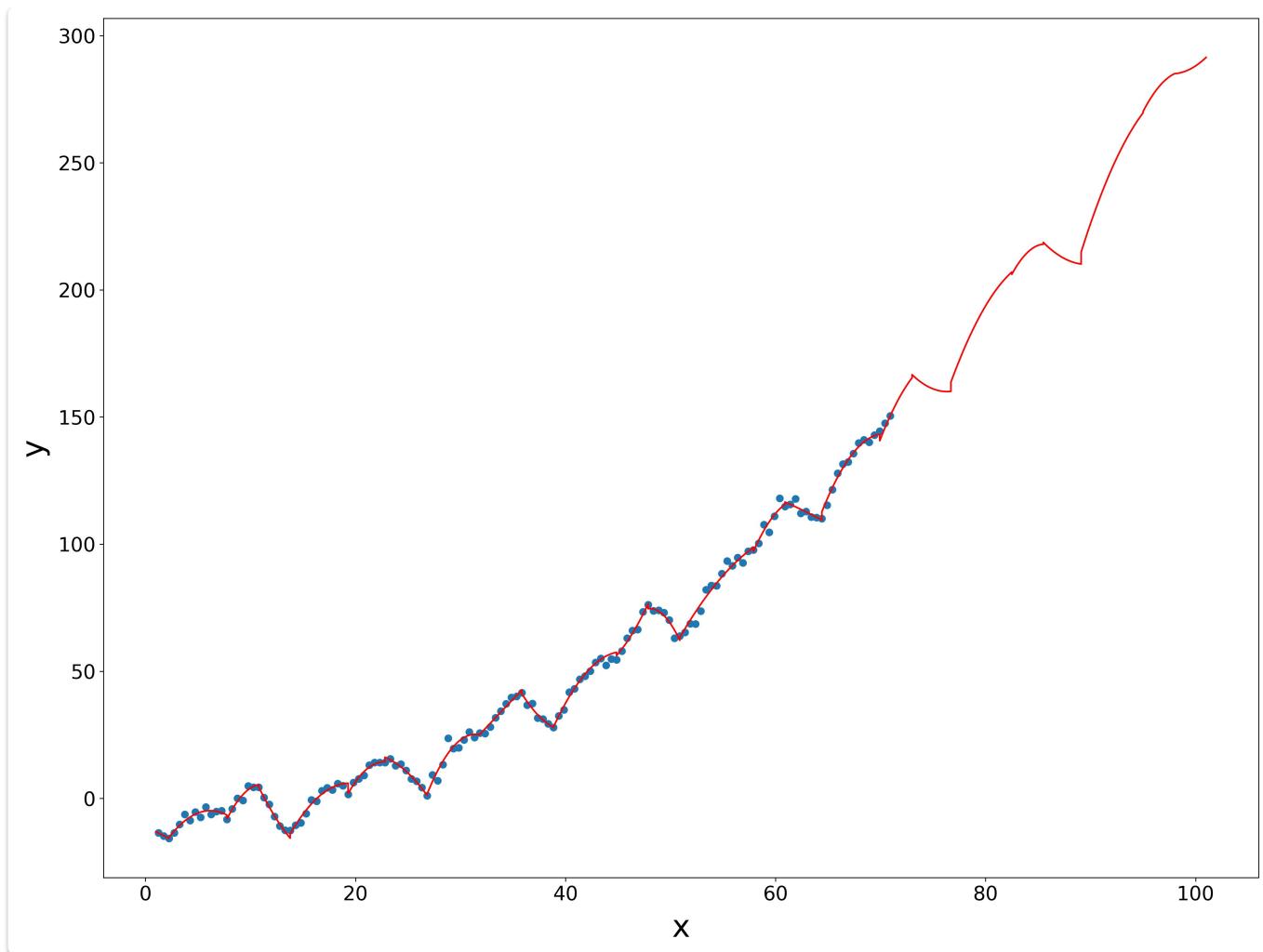
현재 함수의 y값이 과도하게 높은 것으로 파악되어 함수의 y값을 더 내려봤다

p	x	y
2->3	74	170
3->1	77	167
1->2	81	200
2->3	86.5	225
3->1	89.5	220
1->2	93.5	260
2->3	99	290
3->1	102	300

```

[-0.85, 129.534842, -4760.97833],
[0.6, -91.6, 3656.066666666666],
[-0.65, 110.9499999999999, -4522.299999999999],
[-1.2, 205.545455, -8583.861264824382],
[0.5, -89.6666667, 4230.055558544444],
[-0.65, 128.95, -6114.362499999999],
[-1.0, 197.954545, -9510.500471539255],
[0.6, -117.266667, 6014.779662203704],

```



[2023-10-21 03:29:20]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.518558

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 73.422938

- Metric: 76.94149600

- Rank: 42

- Score: 75.0

여전히 크게 개선되는 모습은 아니다

중간 점검

현재 상황

현재 내 점수는 다음과 같다

```
[2023-10-21 03:29:20]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 3.518558
```

```
[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 73.422938
```

- Metric: 76.94149600
- Rank: 42
- Score: 75.0

사실 뒤에다 일차함수 때려박는거보다 점수가 더 안나오고 있다

다른 방법을 생각해보던가 해야 할 거 같은데

다른 사람들 점수

10-21 03:30 기준 다른 사람들의 점수는 다음과 같다

1등

```
- [ASSIGNMENT_01 (2023-10-15 08:04:37)] 20203806 has submitted..! (![💡]Metric: 2.81517e+00 ![🏅]Rank: 1 ![🔥]Score: 100.0)
```

Metric : 2.81517

5등

```
- [ASSIGNMENT_01 (2023-10-18 19:28:40)] 20196151 has submitted..! (![💡]Metric: 7.65482e+00 ![🏅]Rank: 5 ![🔥]Score: 100.0)
```

Metric : 7.65482

1등을 하려면...

Metric 계산 방식은 문제 1의 Metric과 문제 2의 Metric을 합산하는 방식이다

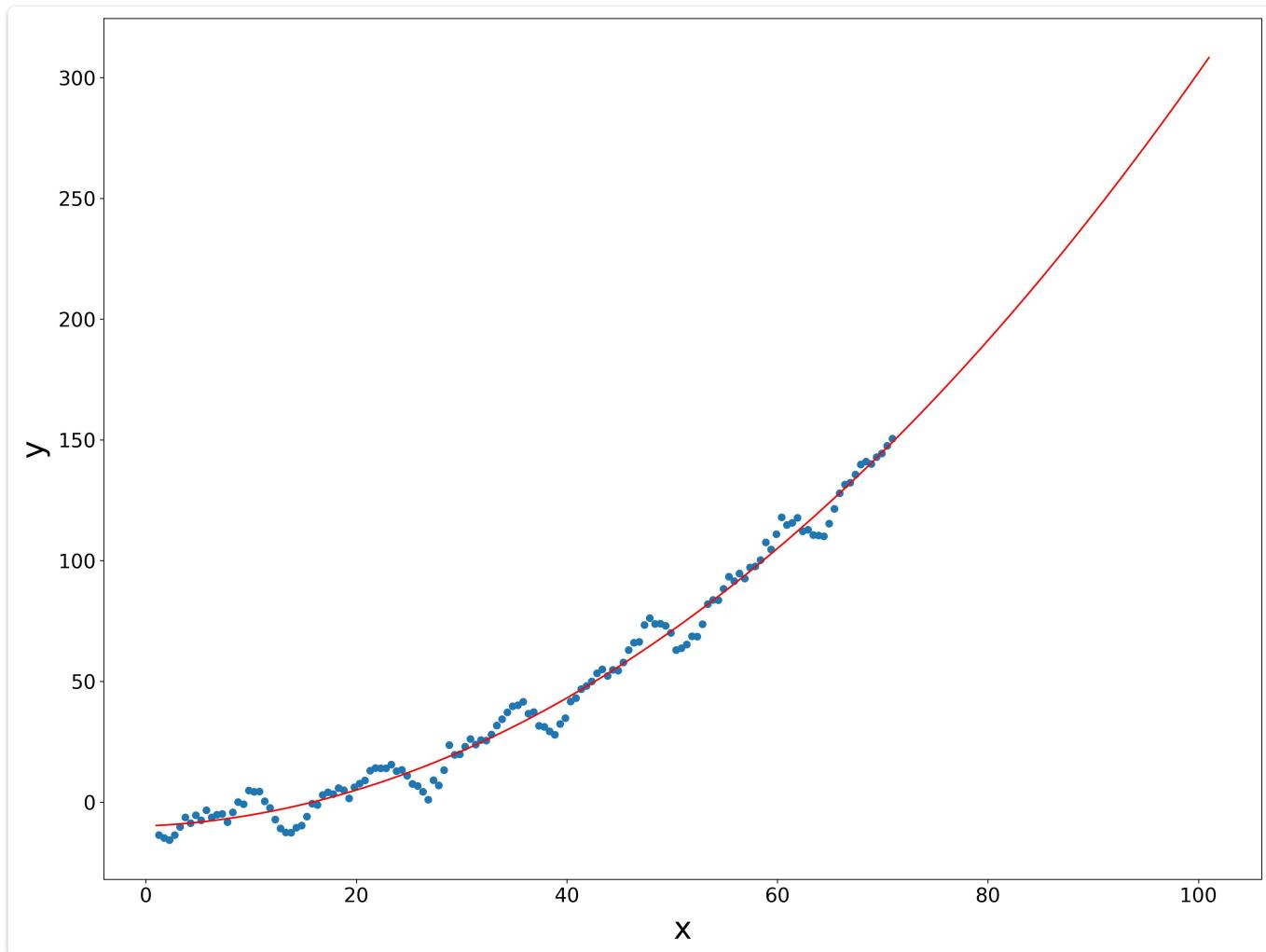
지금 1등의 두 문제 합산 Metric은 내 문제 1의 Metric보다 작다

1등을 하려면 처음부터 다시 해야 된다는 뜻이다

몇 가지 실험

구간을 나누지 않고 poly를 돌려버리면?

3차 함수로 poly를 돌려버리면



[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 31.382493

[SECURITY] Checking security issues has passed!

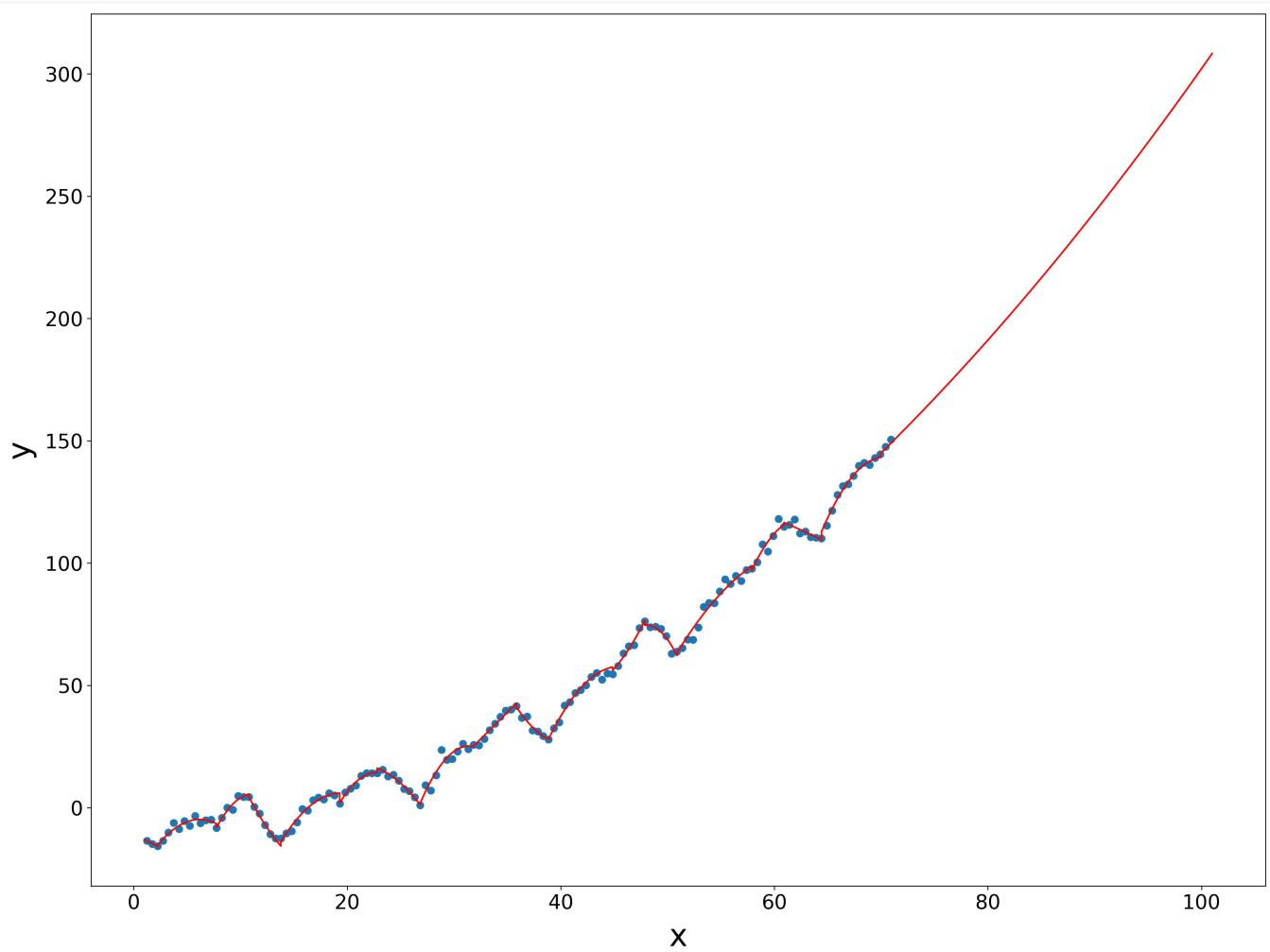
[PROBLEM_02] Testing unseen data

Evaluation metric: 34.006545

- Metric: 65.38903800
- Rank: 33
- Score: 82.5

무려 여태 내가 짠 거보다 점수가 높게 나온다

일단은 문제 1의 metric이 내 함수가 너 낮으므로 둘을 합쳐본다



[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.561504

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 34.006545

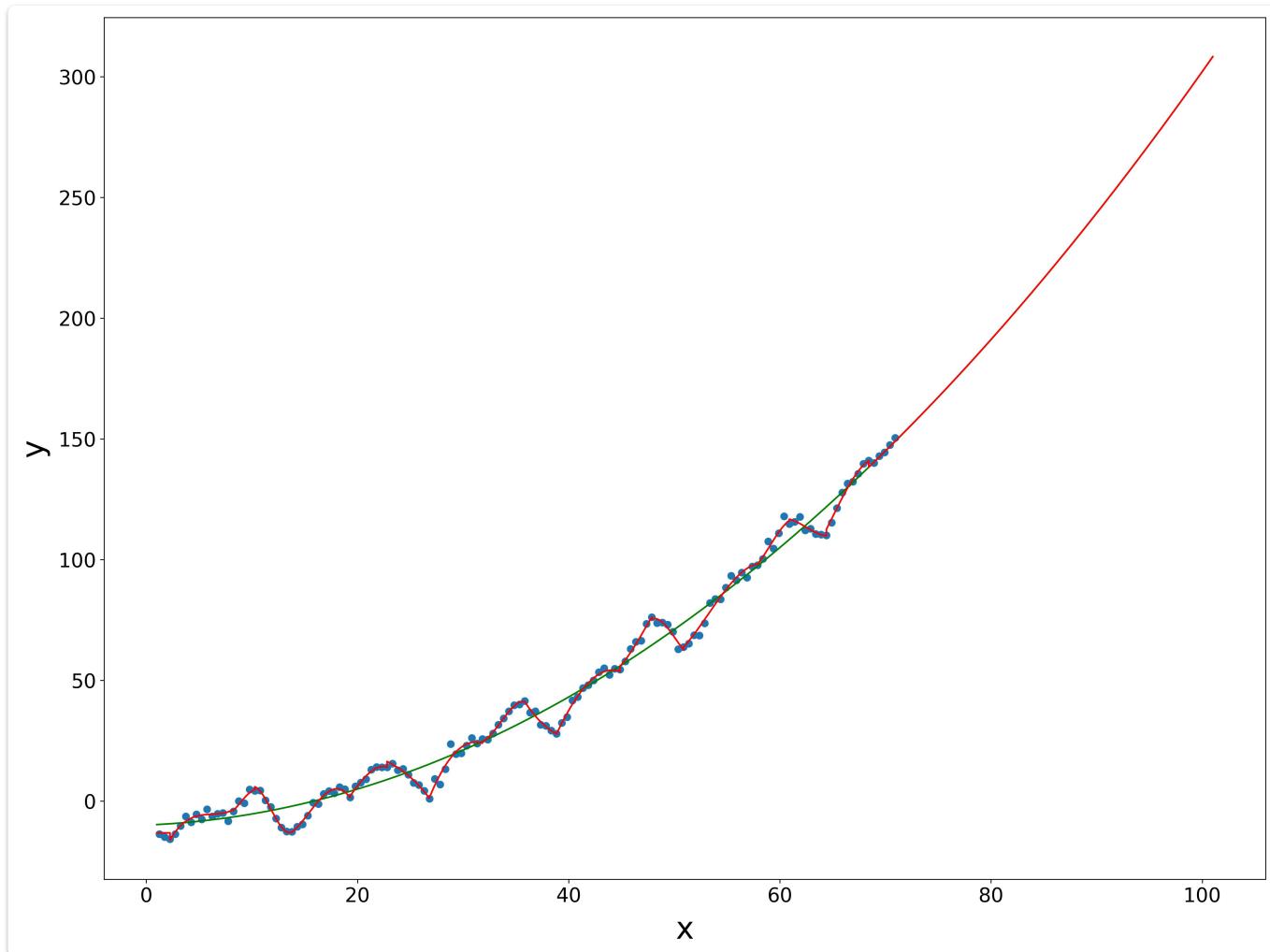
- Metric: 37.56804900

- Rank: 20
- Score: 90.0

일단은 20등...

여러 가지 시도

문제1의 구간을 3차함수로 변경하고 다듬었을 땐



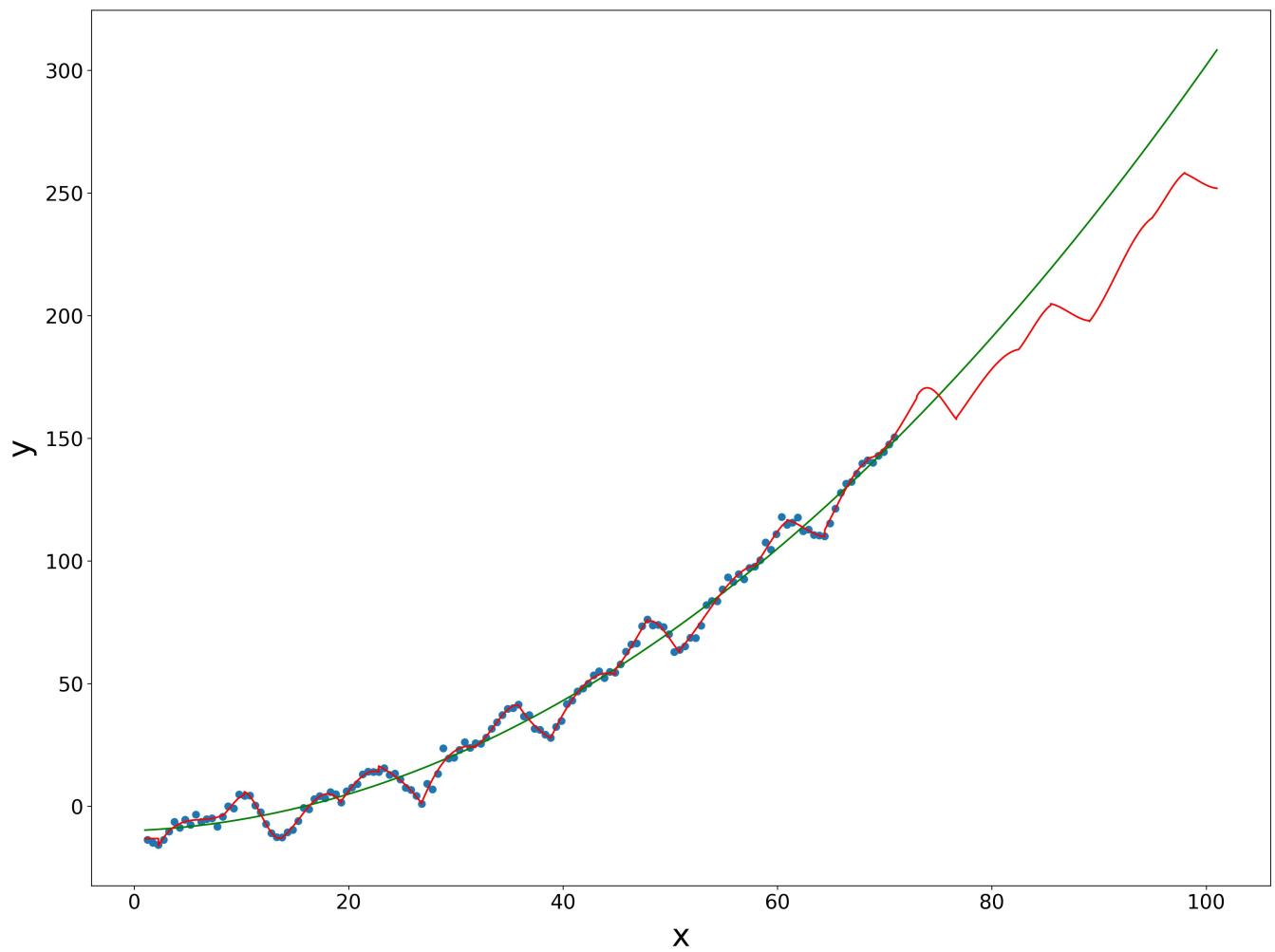
```
[2023-10-21 05:10:59]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 3.454575
```

```
[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 34.006545
```

- Metric: 37.4612000
- Rank: 20
- Score: 90.0

조금 줄었다

앞의 함수를 뒤에 그대로 붙였을 땐



[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.307006

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

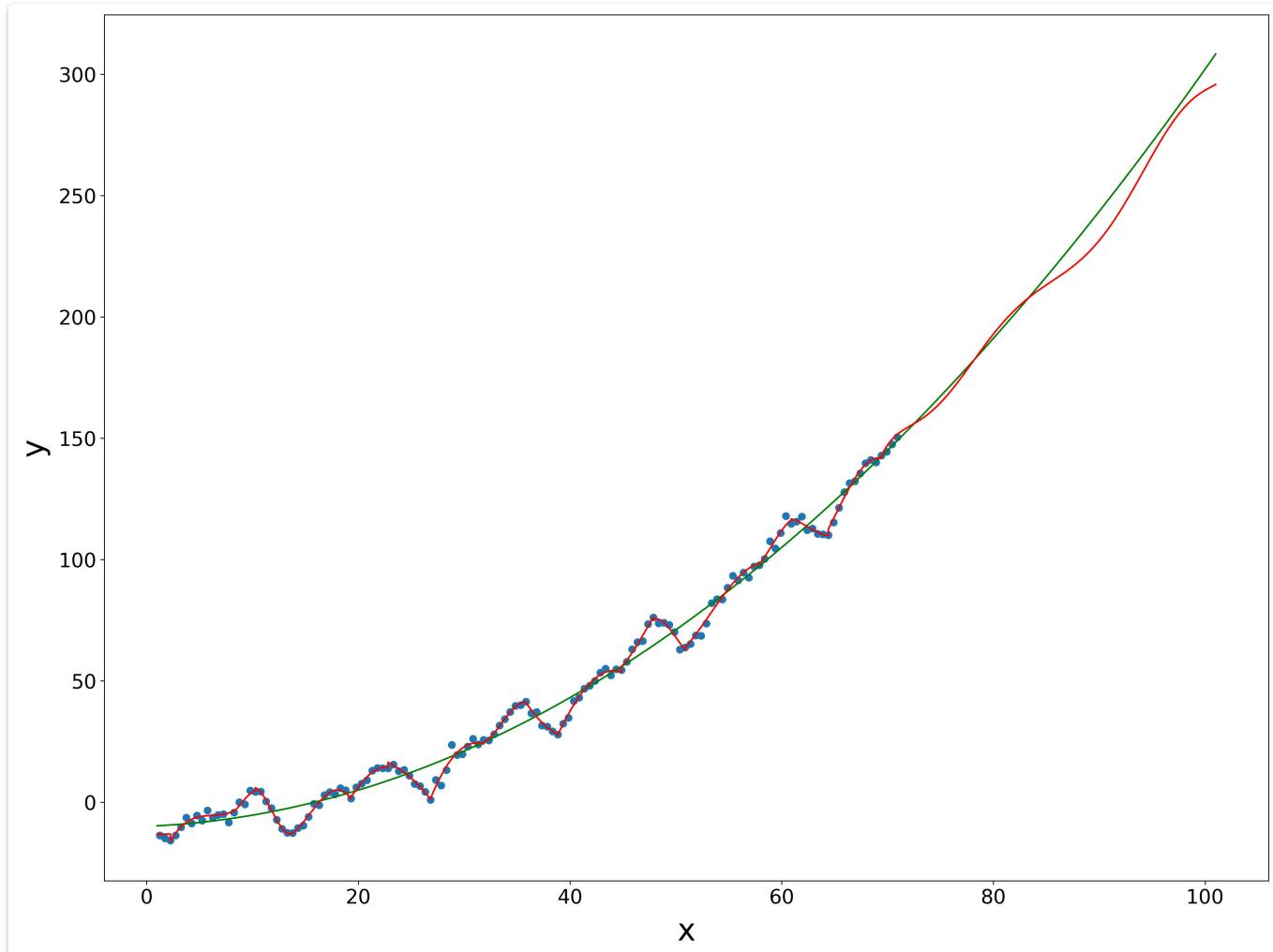
Evaluation metric: 695.377464

- Metric: 698.68447000

- Rank: 47
- Score: 72.5

3차 함수에 근사해야 되는 것으로 보인다

이전에 임의로 생성한 점들을 피팅했을 때



[2023-10-21 06:09:15]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 3.350499

[SECURITY] Checking security issues has passed!

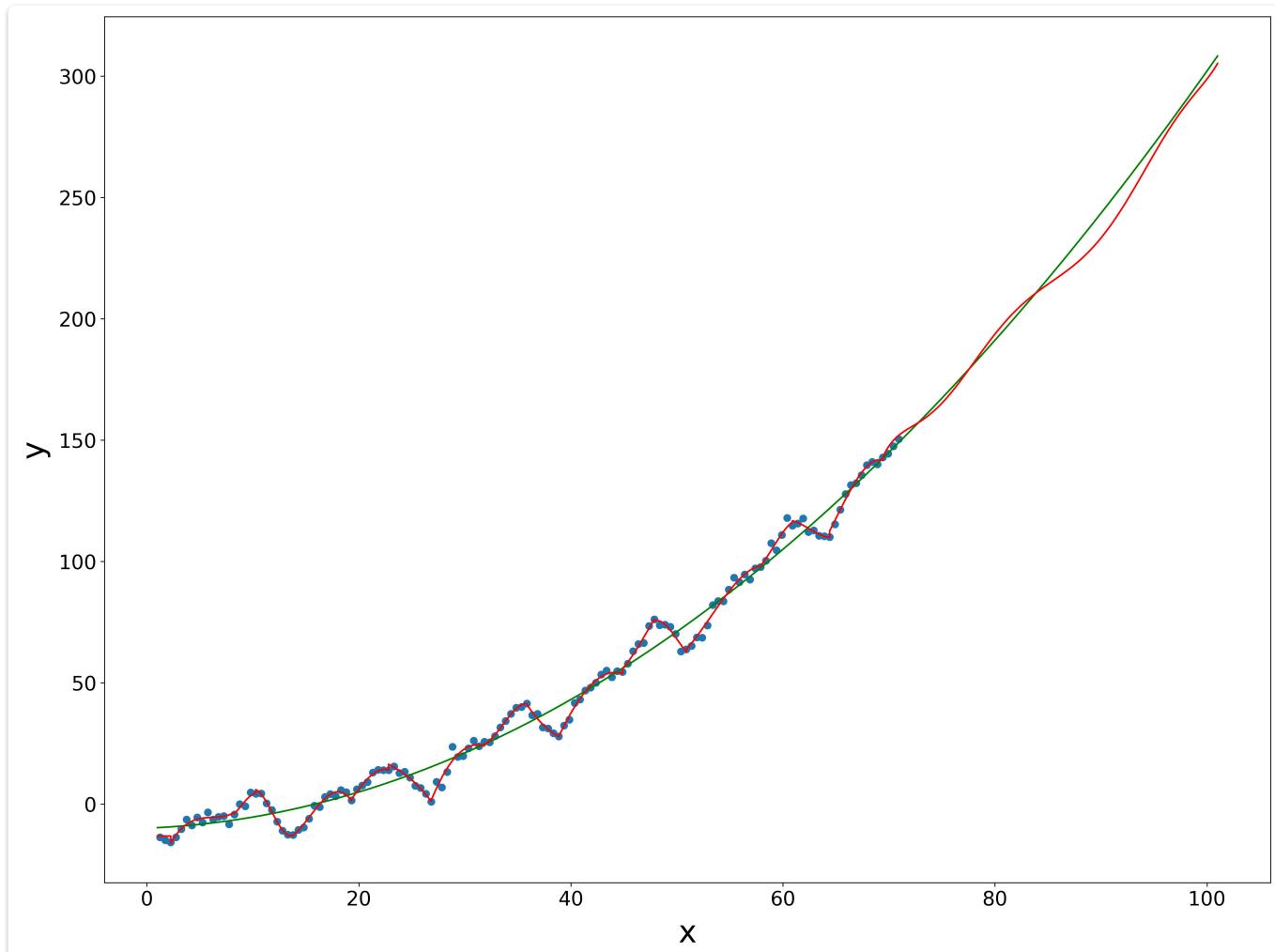
[PROBLEM_02] Testing unseen data

Evaluation metric: 65.103023

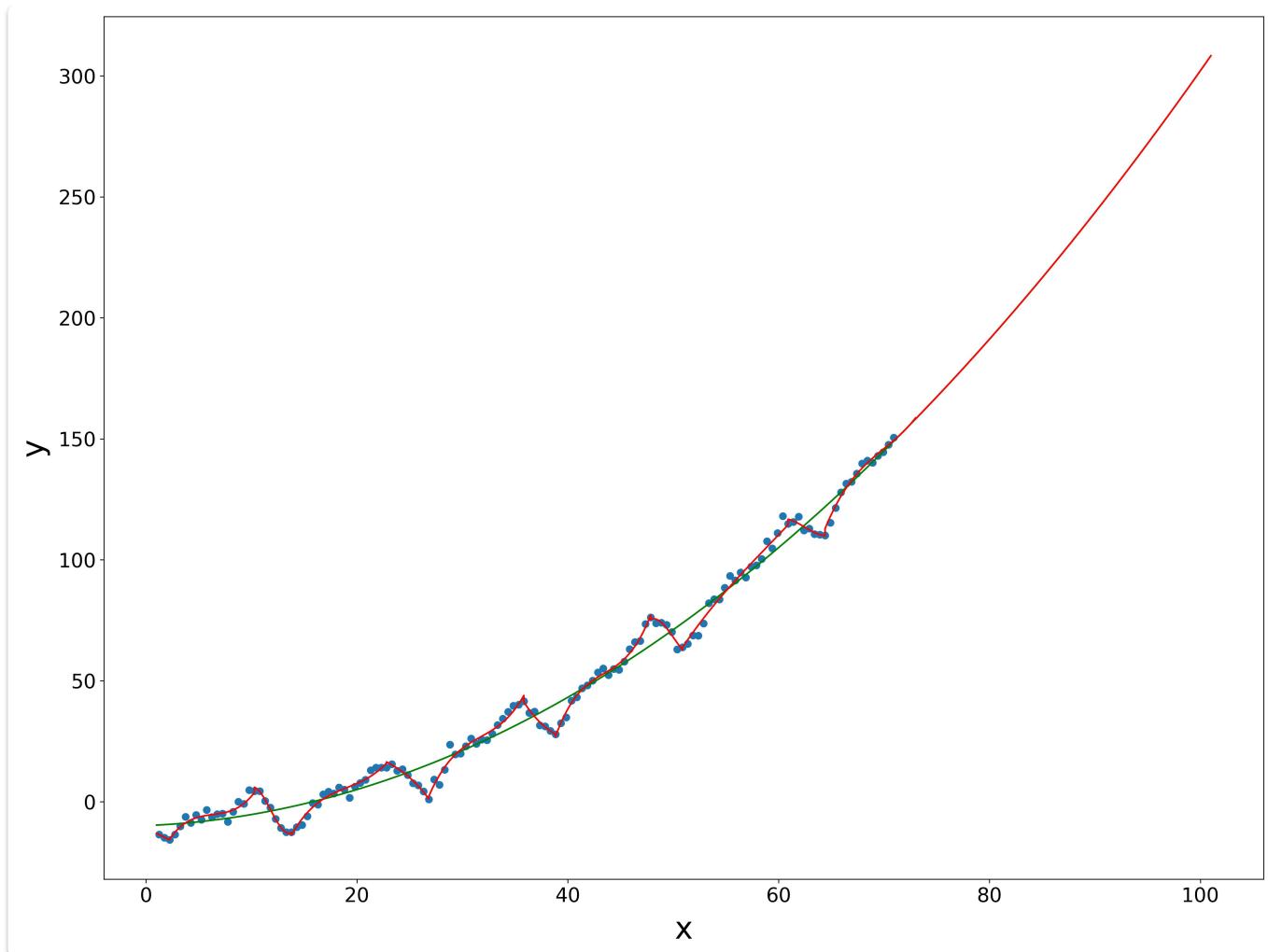
- Metric: 68.45352200

- Rank: 38
- Score: 77.5

metric이 증가했다



패턴을 2개로 줄여버리면?



[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 4.380869

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 33.950962

- Metric: 38.33183100

- Rank: 21

- Score: 90.0

Metric이 상승했으므로 3개 구간으로 나눈 패턴이 더 정확했던 것으로 보인다

새 시작

지금까지 얻은 결론

현재 주어진 데이터로 구간을 예측해 모델을 생성하는 것은 좋은 방법이 아니다

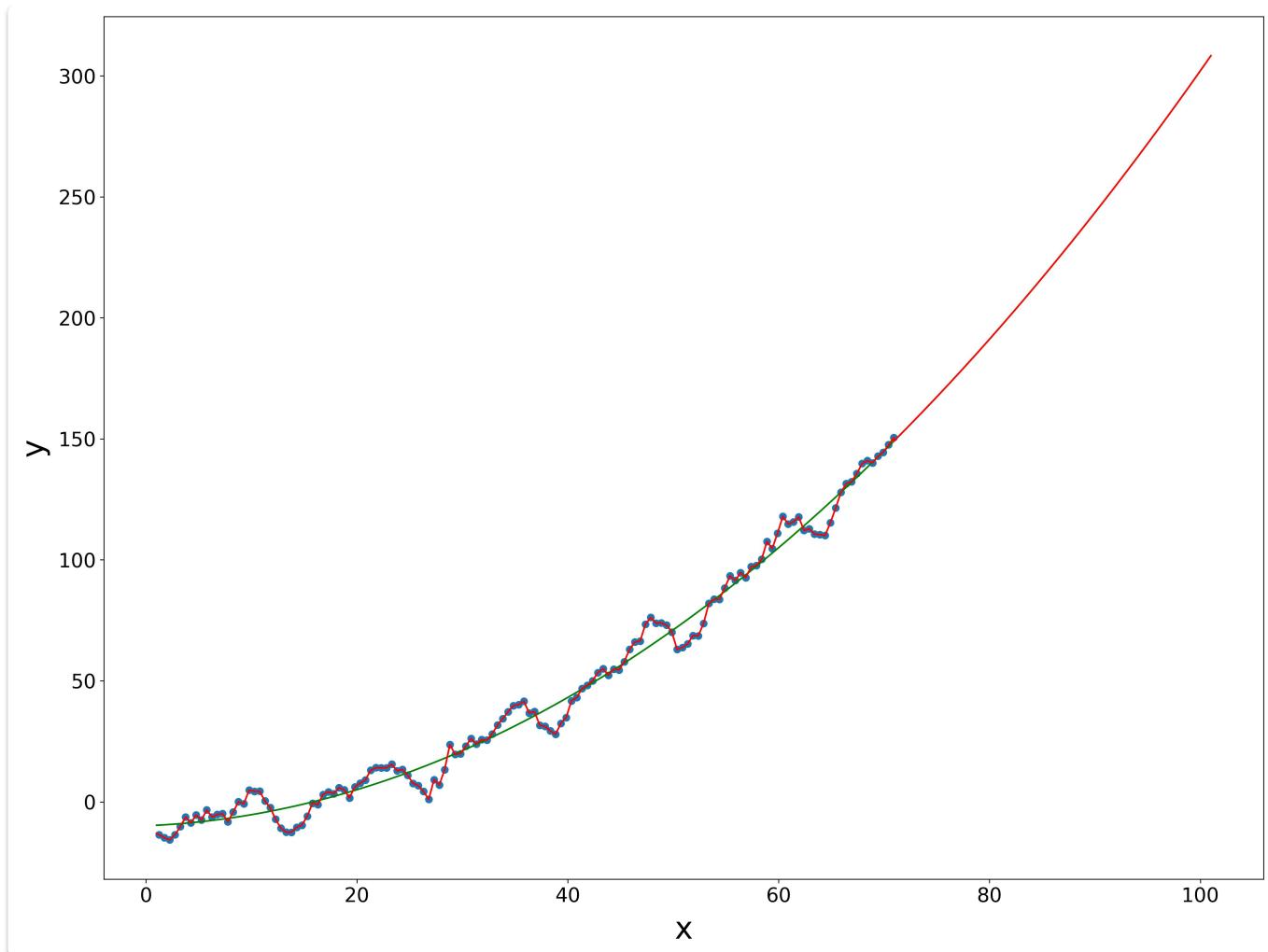
- 생각보다 오차가 드라마틱하게 줄어들지 않음
- 이후 구간을 예측할 수 없음

현재 데이터를 정확하게 피팅할 수 있는 새로운 모델이 필요하다

몇 가지 실험

아예 훈련데이터의 인접한 두 점을 직선으로 이어버리면?

```
def make_linear_function(idx):  
    p = test_data[idx - 1]  
    q = test_data[idx]  
    a = [[p[0], 1], [q[0], 1]]  
    b = [p[1], q[1]]  
    n = np.dot(np.linalg.inv(a), b)  
    return n  
  
def func2(x):  
    y = 1.07963682e-05 * x ** 3 + 2.83970127e-02 * x ** 2 + 1.72070775e-01 * x - 9.87373222e+00  
    return y  
  
def func(x):  
    y = []  
  
    for i in x:  
        n = 1  
        while(True):  
            if n >= len(test_data):  
                y.append(func2(i))  
                break  
            elif i < test_data[n][0]:  
                f = make_linear_function(n)  
                y.append(f[0] * i + f[1])  
                break  
            else:  
                n += 1  
                continue  
  
    y = np.array(y)  
    return y
```



[2023-10-21 23:03:45]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.323682

[SECURITY] Checking security issues has passed!

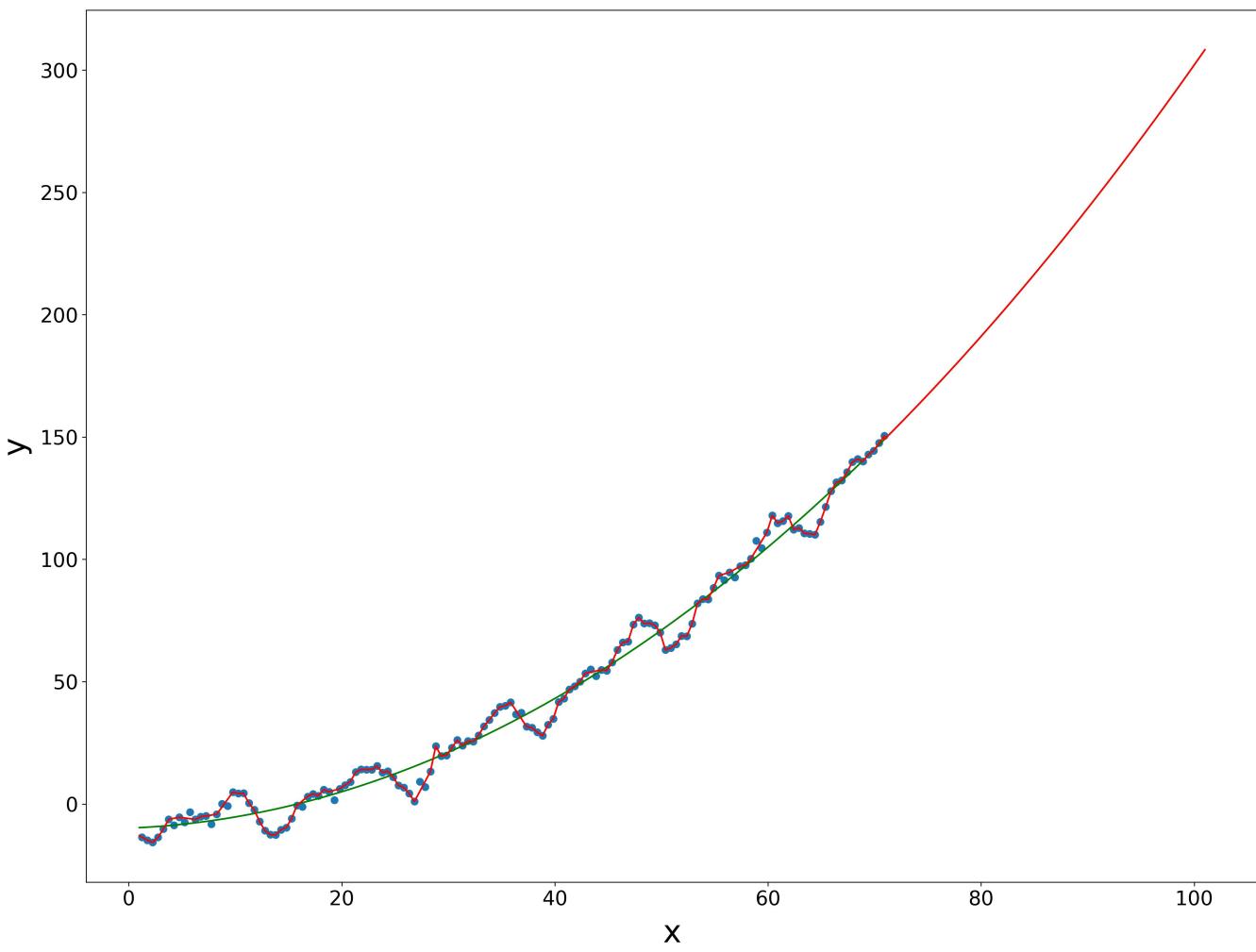
[PROBLEM_02] Testing unseen data

Evaluation metric: 34.006545

- Metric: 36.33022700
- Rank: 21
- Score: 90.0

이래도 문제 1의 metric이 2.32로 줄었다

같은 방식으로 함수를 조금 더 다듬었다



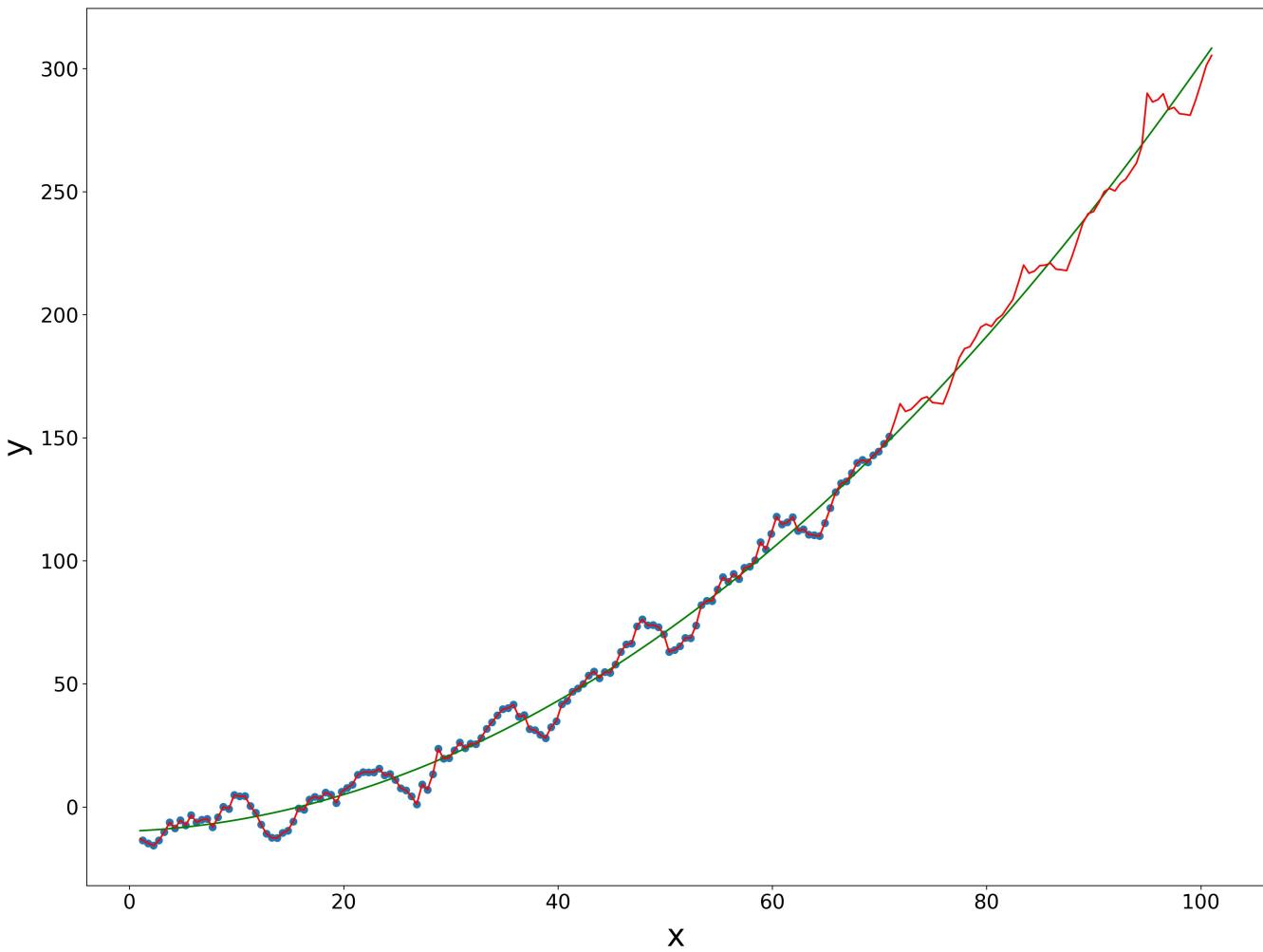
```
[2023-10-22 00:31:00]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 3.032857
```

```
[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 34.006545
```

- Metric: 37.03940200
- Rank: 21
- Score: 90.0

안다듬는게 좋겠다

위의 일차식을 뒤에다 잘 붙여놓으면



[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.210297

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

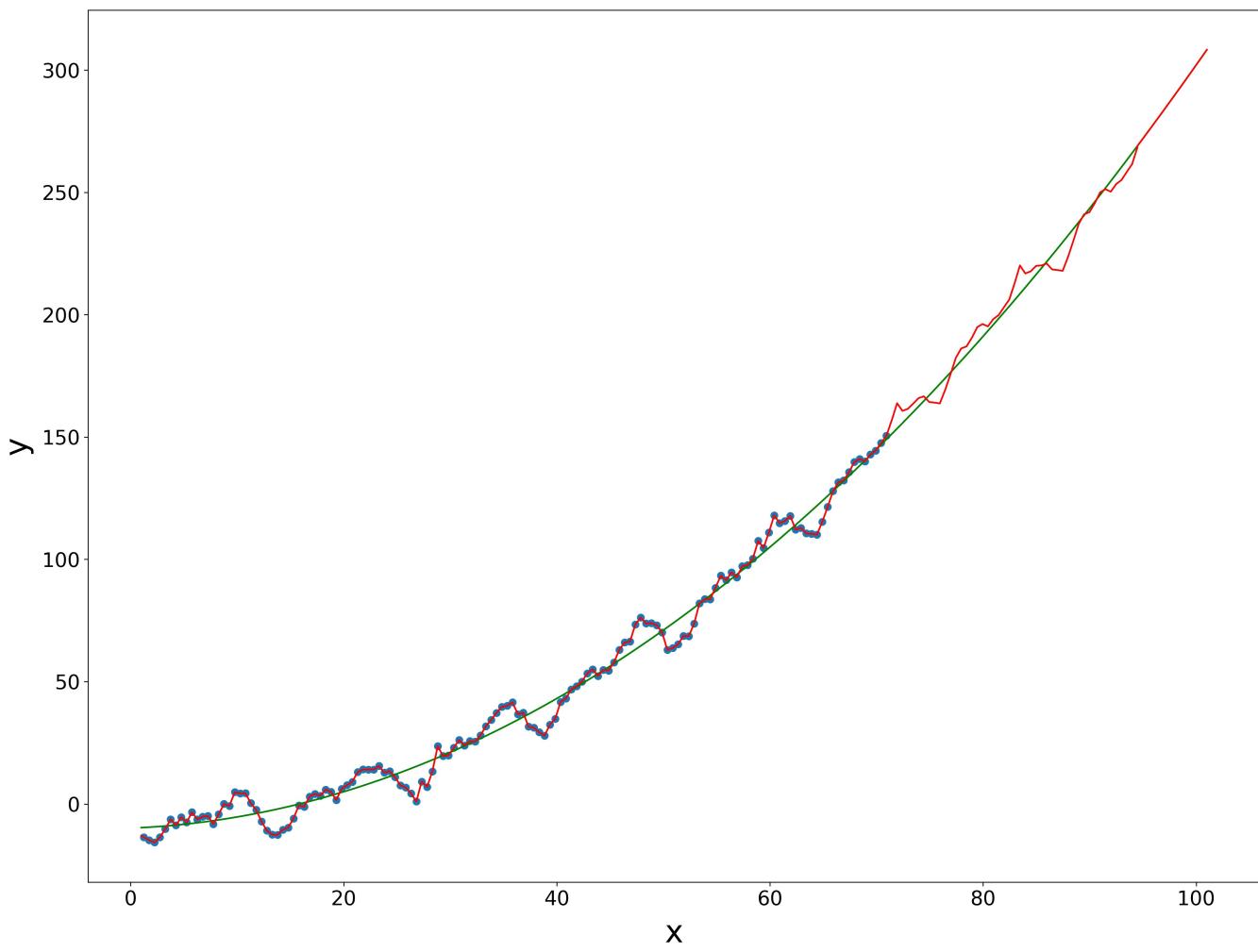
Evaluation metric: 66.009396

- Metric: 68.21969300

- Rank: 38

- Score: 77.5

끝부분이 약간 하자있는거같은데 끝부분을 한번 없애보자



[2023-10-22 01:32:42]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.210297

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 39.518992

- Metric: 41.72928900
- Rank: 23
- Score: 87.5

앞부분에도 하자가 있는 것으로 보인다

x	y	x 증가량	y 증가량
2.2556390977443606	-15.703351869377082	-	-

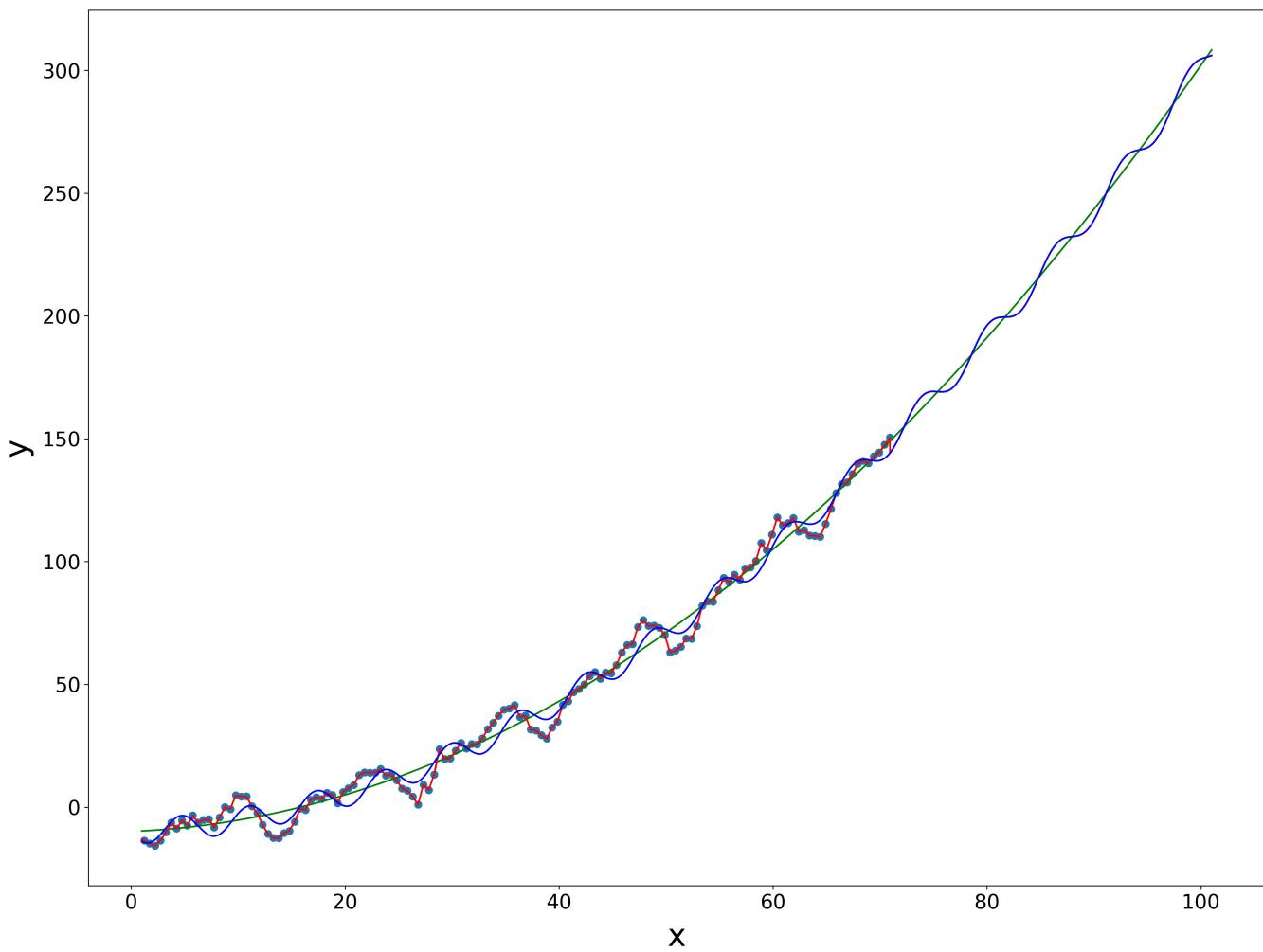
x	y	x 증가량	y 증가량
10.275689223057643	4.332933182249585	8.0200501253132824	20.036285051626667
13.784461152882205	-12.60412206710894	3.508771929824562	-16.937055249358525
23.308270676691727	15.537777888898923	9.523809523809522	28.141899956007863
26.81704260651629	1.029043543439636	3.508771929824563	-14.508734345459287
35.83959899749373	41.55890040439935	9.02255639097744	40.529856860959714
38.847117794486216	27.899441476292964	3.007518796992486	-13.659458928106386
47.86967418546366	76.18294674859914	9.022556390977444	48.283505272306176
50.37593984962406	62.95947126439767	2.5062656641604	-13.22347548420147
61.9047619047619	117.7783283452736	11.52882205513784	54.81885708087593
64.41102756892231	110.0894285872276	2.50626566416041	-7.688899758046

문득 든 생각

이거 3차함수에 사인함수를 더한 거 아니야?

```
def func2(x):
    y = 1.07963682e-05 * x ** 3 + 2.83970127e-02 * x ** 2 + 1.72070775e-01 * x - 9.87373222e+00
    return y

def func3(x):
    y = func2(x) - 5 * np.sin(x)
    return y
```



[2023-10-22 02:12:06]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.715325

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 24.203352

- Metric: 26.91867700
- Rank: 17
- Score: 92.5

이거 좀만 더 다듬으면 뭔가 될 거 같은데

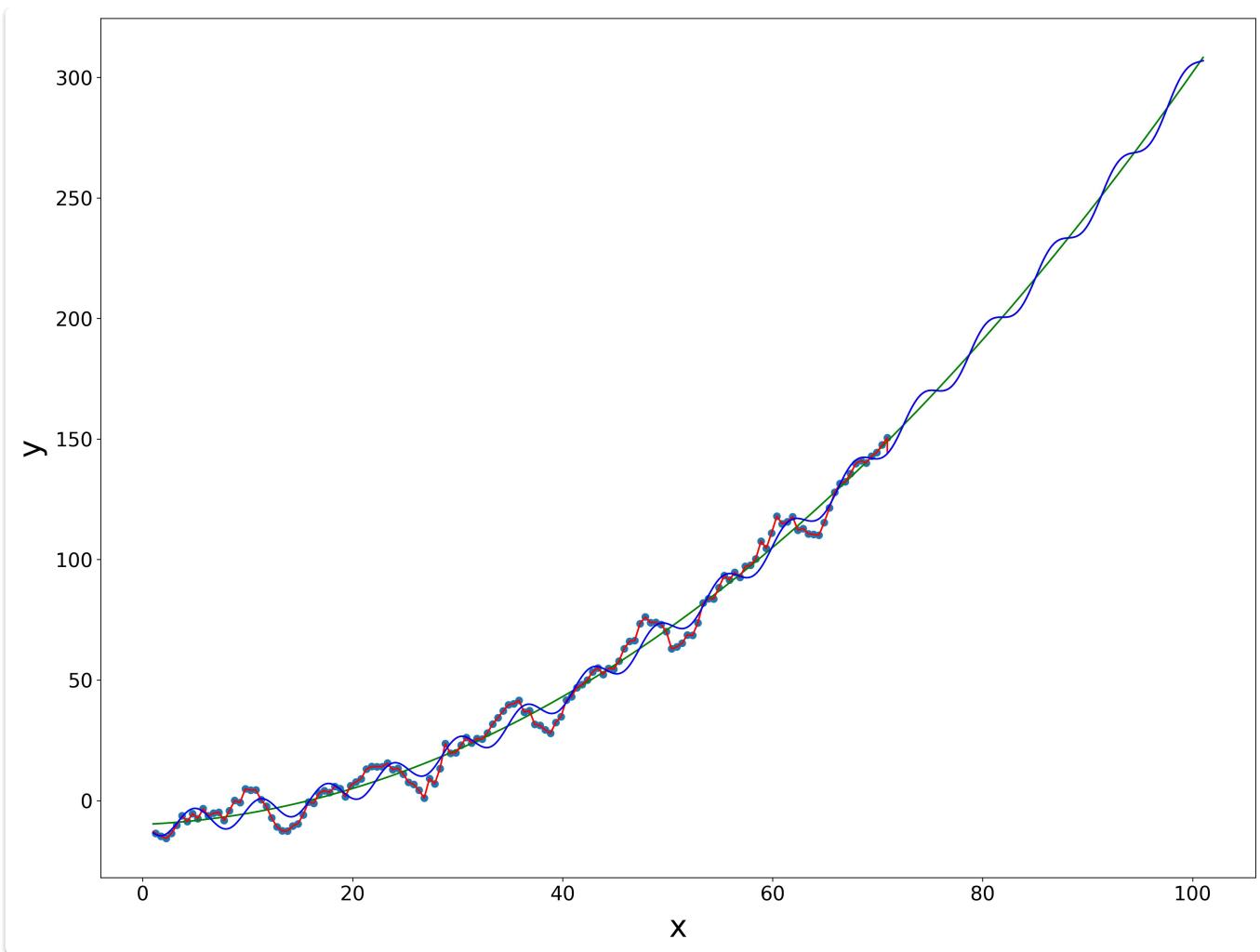
```
def func2(x):
    y = 1.07963682e-05 * x ** 3 + 2.83970127e-02 * x ** 2 + 1.72070775e-01 * x - 9.87373222e+00
    return y

def func3(x):
    y = func2(x) - 5 * np.sin(x) + np.cos(x)
    return y
```

[2023-10-22 02:23:34]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 2.762093

[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 26.993873

- Metric: 29.75596600
- Rank: 17
- Score: 92.5



이런 식으로 함수를 피팅할 수 있을 거 같은데?

3번째 접근

아이디어

훈련 데이터가 어떻게 생성됐는가

-> 특정한 함수에서 적당한 오차로 랜덤값을 뽑아서 데이터를 추출했을 것

적당한 함수 개형을 찾아서 피팅하기만 해도 충분히 좋은 점수가 나올 수 있을 것이다

그렇다면 어떻게 해결할 수 있을까?

- 적당한 함수 개형을 찾는다
- `scipy` 라이브러리의 `curve_fit`을 사용해 함수를 피팅한다

적절한 함수 개형 찾기

계속 찾아보자

```

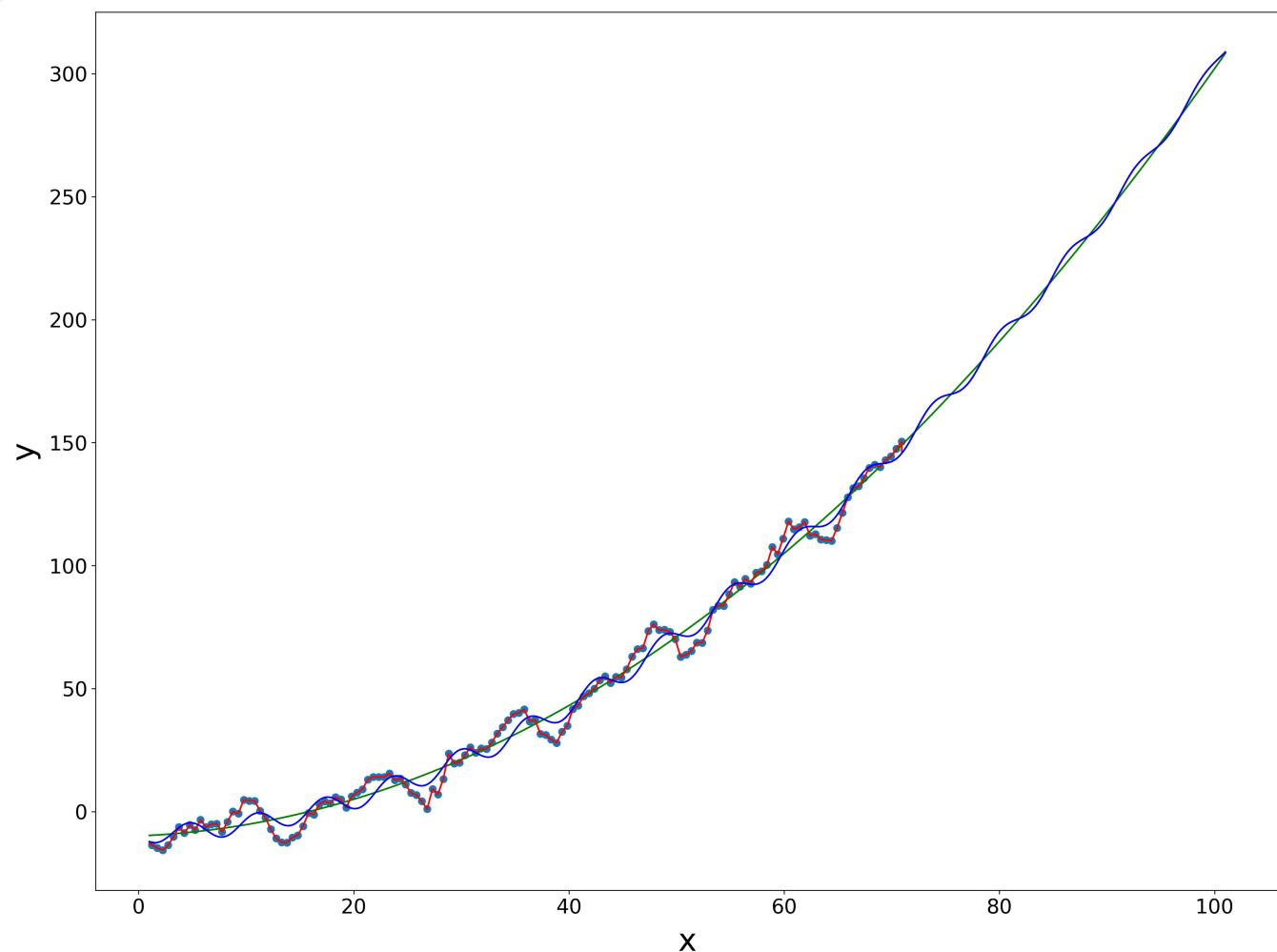
from scipy.optimize import curve_fit

x = data[:, 0]
y = data[:, 1]

def quadratic_model(x, a, b, c, d, e, f, g):
    return a * x ** 3 + b * x ** 2 + c * x + (d * x ** 2 + e * x + f) * np.sin(x) + g

params, covariance = curve_fit(quadratic_model, x, y)

```



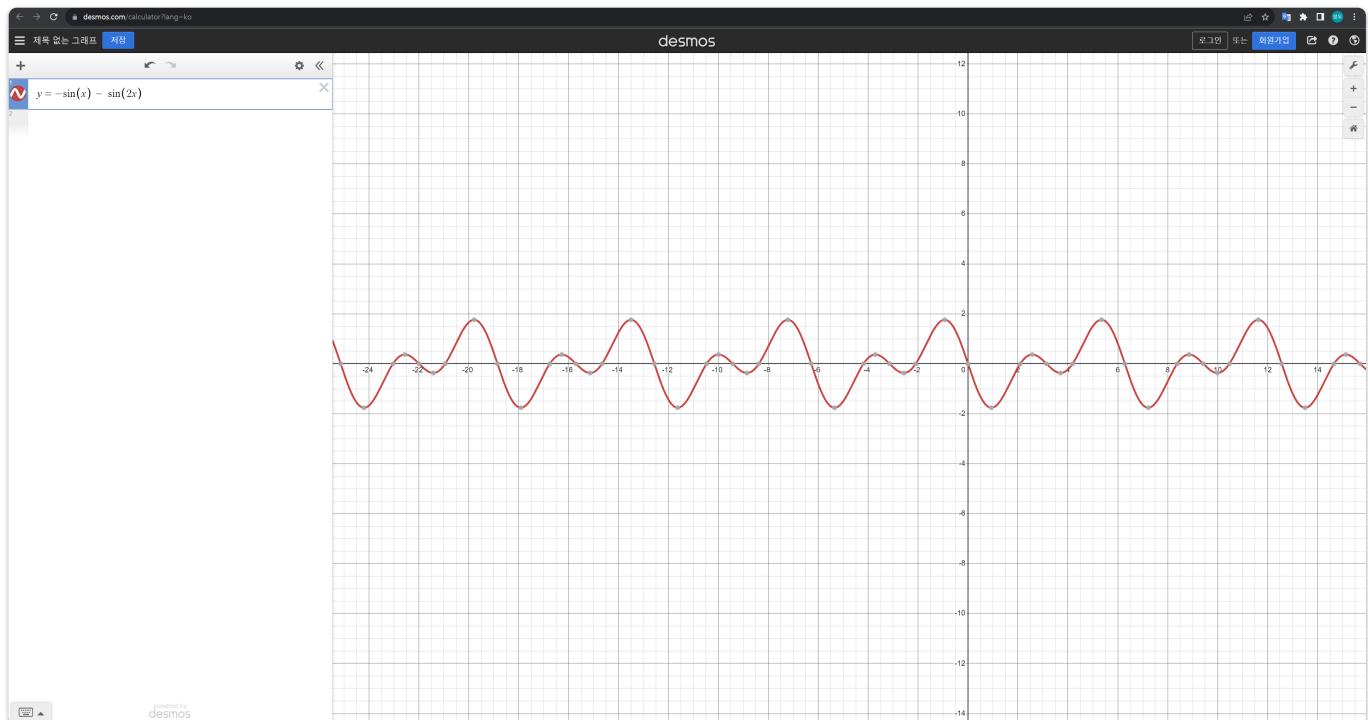
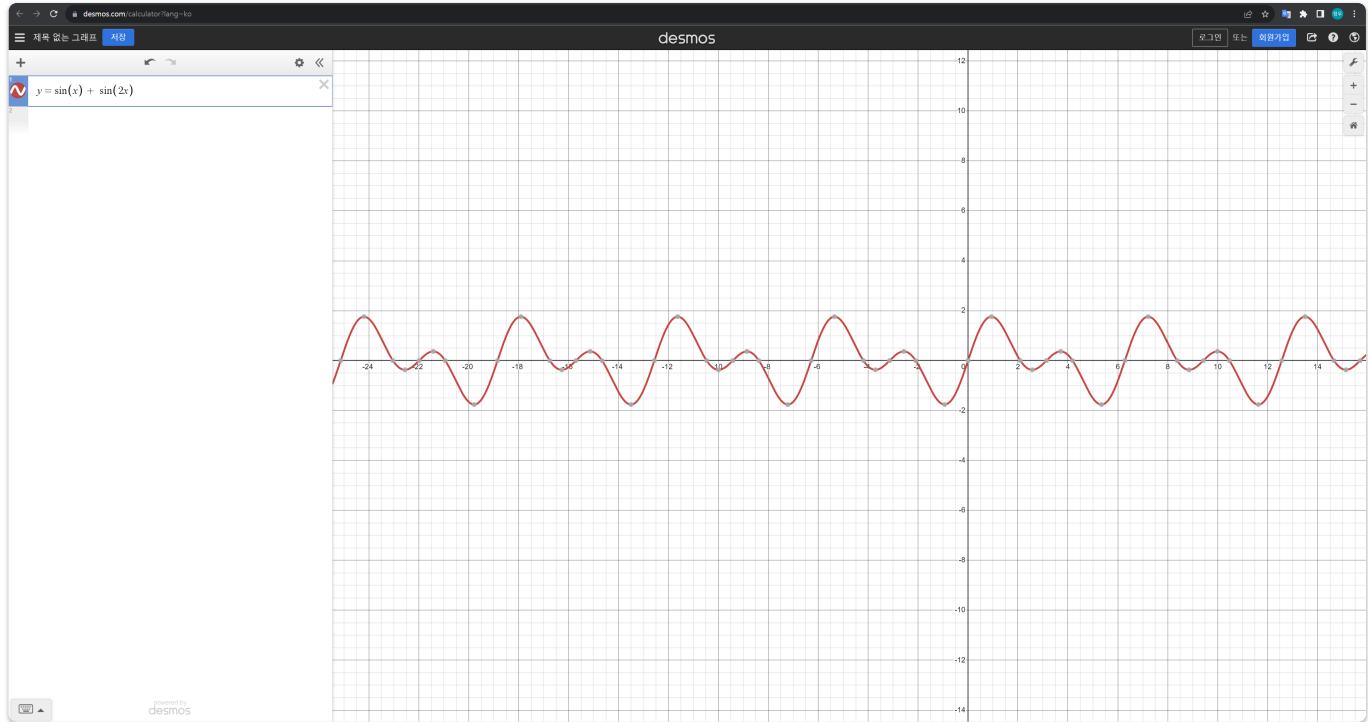
[2023-10-22 03:55:04]
[SECURITY] Checking security issues has passed!
[PROBLEM_01] Testing observed data
Evaluation metric: 2.558908

[SECURITY] Checking security issues has passed!
[PROBLEM_02] Testing unseen data
Evaluation metric: 23.230880

- Metric: 25.78978800
- Rank: 16
- Score: 92.5

아무리 봐도 삼각함수의 합성같은데 어떻게 합성해야 나올지 모르겠어서

함수 그래프 계산기 사이트에서 어떻게 합성해야 되는지 계속 실험했다



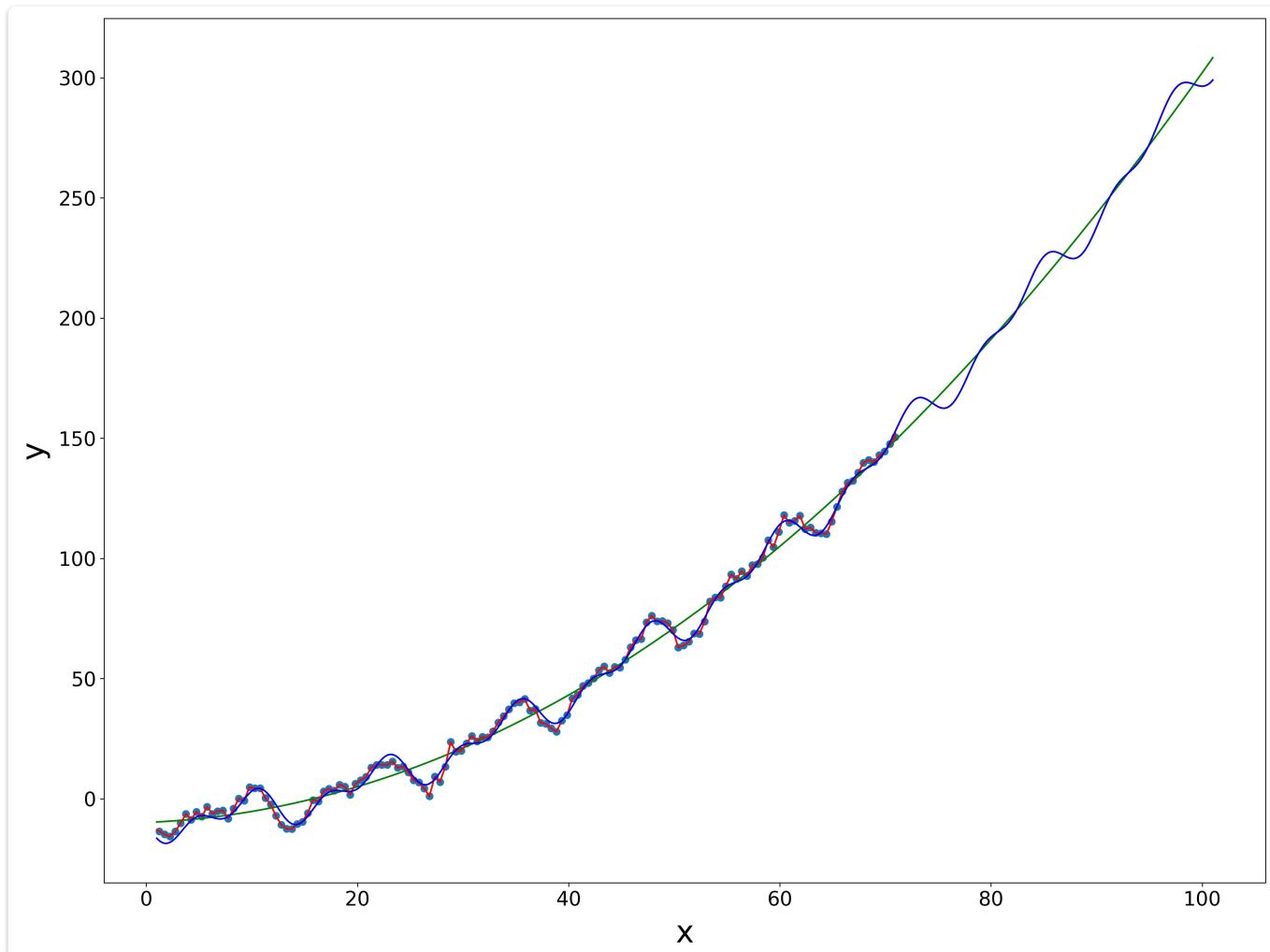
원하는 모양을 찾았다.

바로 함수에 대입해봤다.

```
def quadratic_model(x, a, b, c, d, e, f, g):
    return a * x ** 3 + b * x ** 2 + c * x + d + e * np.sin(f * x) + g * np.sin(2 * f * x)
```

```
def func3(x):
    y = 1.0214458575150344e-05 * x ** 3 + 0.02844802434719404 * x ** 2 + 0.17095659922519982 * x
    + -9.86967380508745 + -5.766219235416621 * np.sin(0.5065138419048528 * x) +
    -4.697246026427157 * np.sin(2 * 0.5065138419048528 * x)
    return y
```

```
1.0214458575150344e-05 * x ** 3 + 0.02844802434719404 * x ** 2 + 0.17095659922519982 * x
+ -9.86967380508745 + -5.766219235416621 * np.sin(0.5065138419048528 * x) +
-4.697246026427157 * np.sin(2 * 0.5065138419048528 * x)
```



[2023-10-24 14:24:58]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.211558

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 10.313945

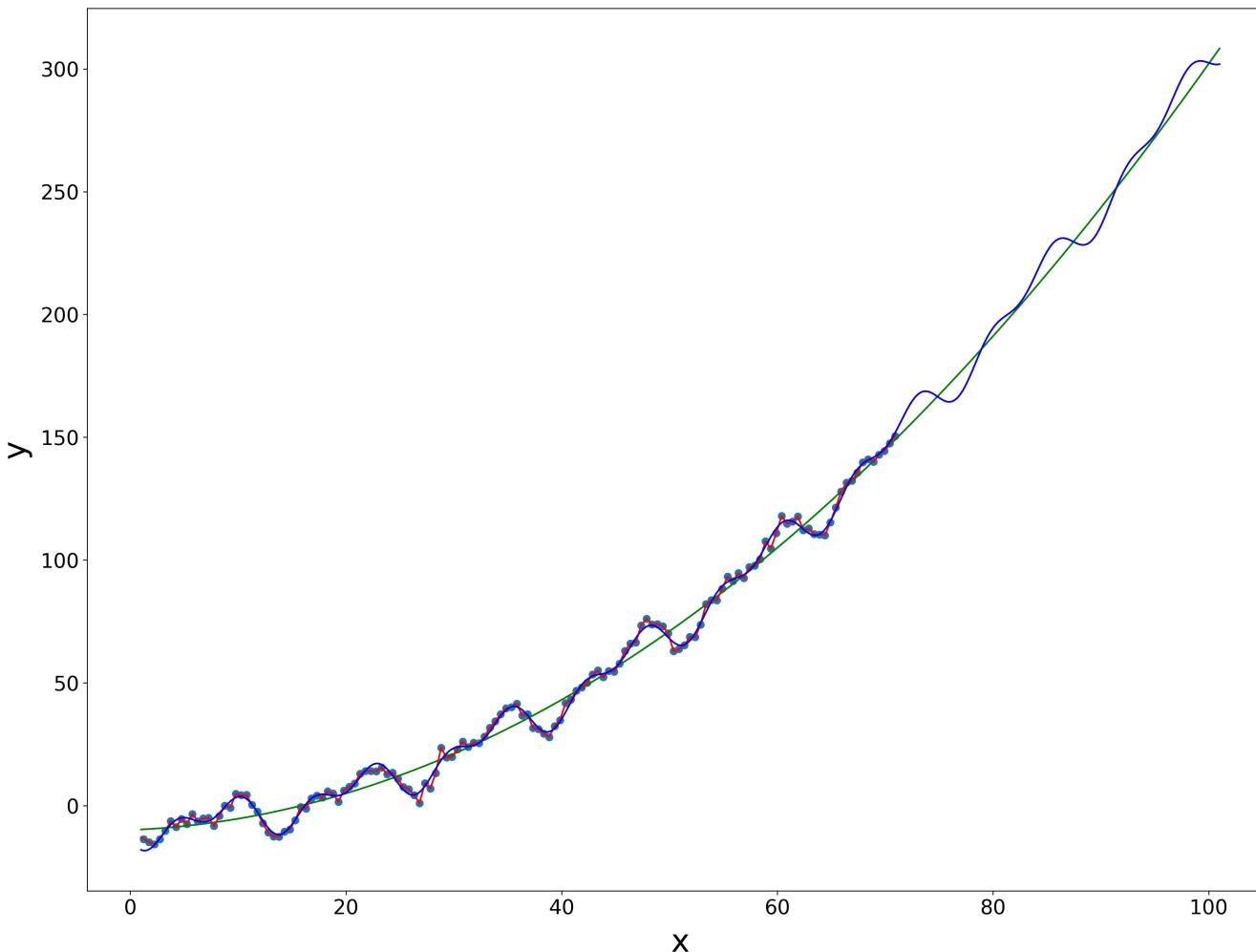
- Metric: 12.52550300
- Rank: 22
- Score: 87.5

마지막날이 되니까 metric 12.5가 22등밖에 안된다
말이 안된다...

3차함수를 2차함수로 줄이면?

```
def quadratic_model(x, b, c, d, e, f, g, h):
    return b * x ** 2 + c * x + d + e * np.sin(f * x + h) + g * np.sin(2 * f * x + h)
```

```
def func3(x):
    # y = -1.0214458575150344e-05 * x ** 3 + 0.028444802434719404 * x ** 2 + 0.17095659922519982 * x + -9.86967389508745 + -5.766219235416621 * np.sin(0.5065138419048528 * x) + -4.6972468626427157 * np.sin(2 * 0.5065138419048528 * x)
    # y = -4.672589716063621e-05 * x ** 3 + 0.035676456070955766 * x ** 2 + -0.000758387537823099 * x + -7.427283509199 + -5.042123708896191 * np.sin(0.50027611199960846 * x + 0.46760602489340961) + -4.7732307926766 * np.sin(2 * 0.50027611199960846)
    y = 0.030599391916395575 * x ** 2 + 0.058893121773313 * x + -8.407762039048782 + -5.892426664318304 * np.sin(0.5000054698264715 * x + 0.48803462287003157) + -4.785135902940669 * np.sin(2 * 0.5000054698264715 * x + 0.48803462287003157)
    return y
```



[2023-10-24 14:38:32]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 2.212061

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 4.206891

- Metric: 6.41895200

- Rank: 6

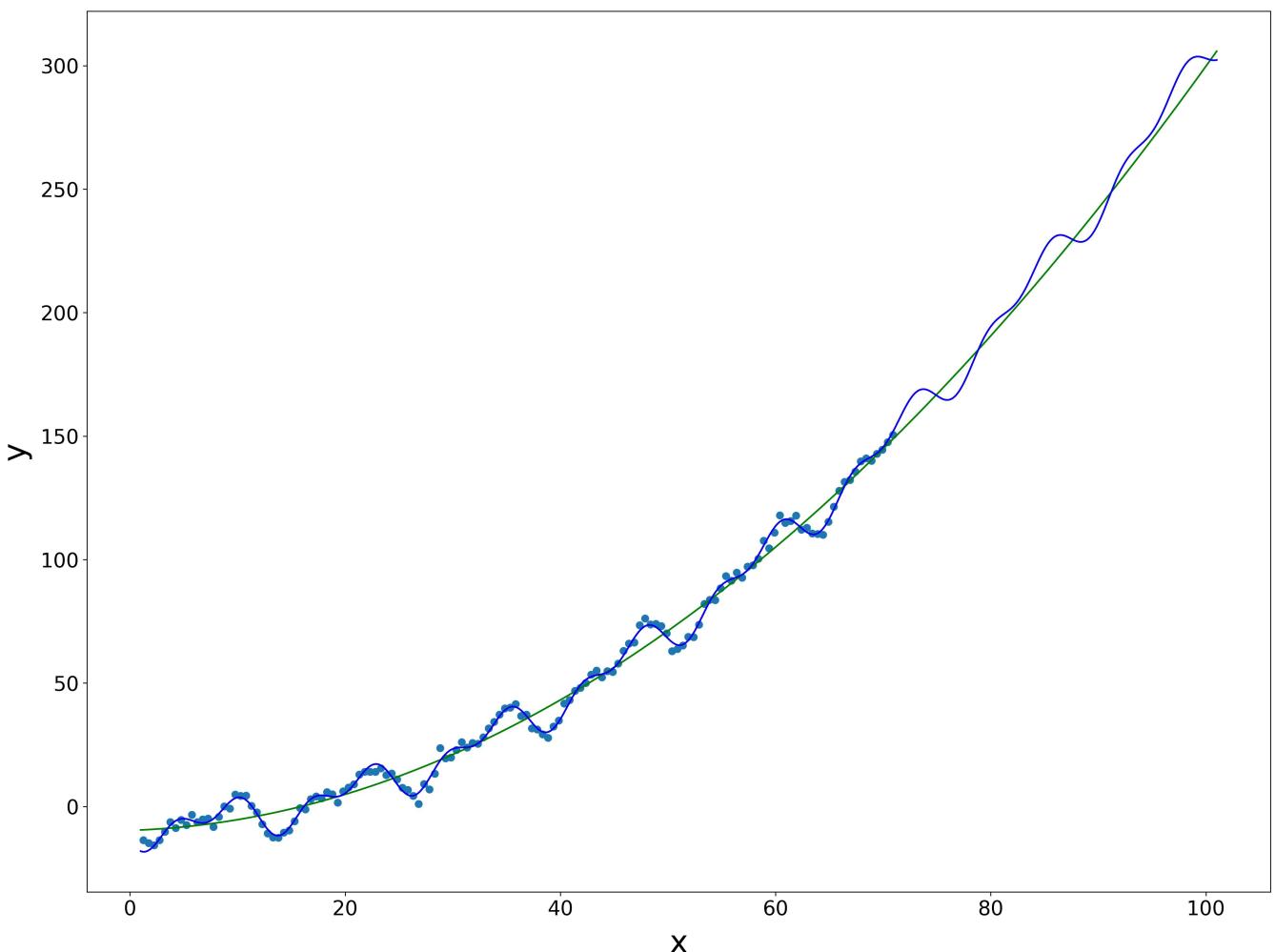
- Score: 97.5

이거다이거야

```
def quadratic_model(x, a, b, c, d, e, f, g, h, i):
    return a * x ** 2 + b * x + c + d * np.sin(e * x + f) + g * np.sin(h * x + i)

params, covariance = curve_fit(quadratic_model, x, y, [0.030599391910395575, 0.058893121773313, -8.407762039048702, -5.892426664318304, 0.5000054698264715, 0.48003462287003157, -4.785135902940669, 2 * 0.5000054698264715, 0.48003462287003157])
print(f"({params[0]} * x ** 2 + {params[1]} * x + {params[2]} + {params[3]} * np.sin({params[4]}) * x + {params[5]}) + {params[6]} * np.sin({params[7]}) * x + {params[8]})")

def func3(x):
    y = 0.030599391910395575 * x ** 2 + 0.058893121773313 * x + -8.407762039048702 -5.892426664318304 * np.sin(0.5000054698264715 * x + 0.48003462287003157) -4.785135902940669 * np.sin(2 * 0.5000054698264715 * x + 0.48003462287003157)
    y = 0.03061328230929357 * x ** 2 + 0.05874566958141142 * x + -8.424235322058871 + -5.901368651249288 * np.sin(0.49928350074009786 * x + 0.4961367913352101) + -4.78163987424188 * np.sin(0.999516076941776 * x + 0.5122538832391688)
    return y
```



[2023-10-24 21:02:08]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 1.485355

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 4.475937

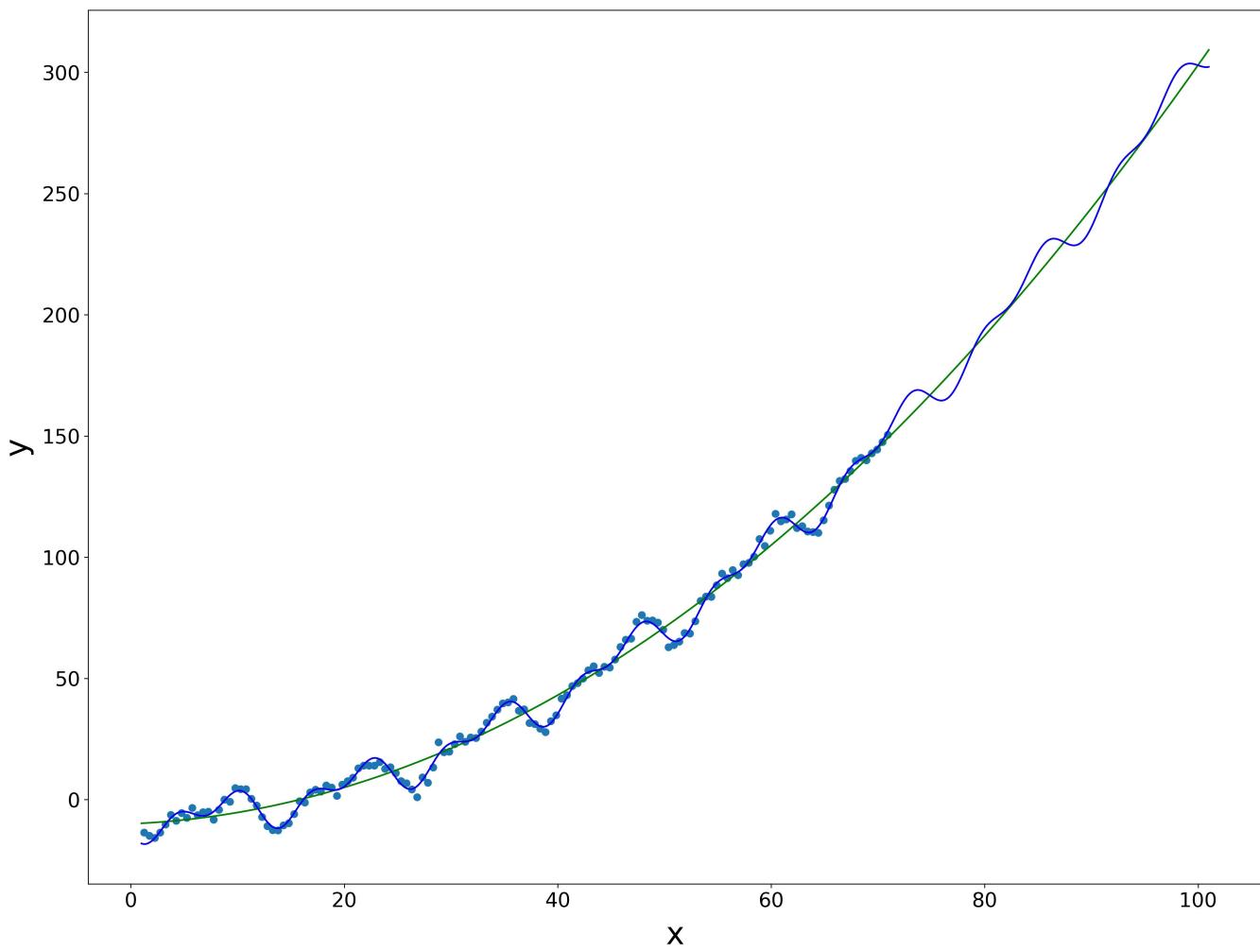
- Metric: 5.96129200

- Rank: 1

- Score: 100.0

추하게 끝부분 노가다로 낮춰 5등 안에 들려 했는데 큰일났다

```
for i in x:
    find = False
    if i < 71.0:
        n = binary_search_in_arr(i)
        if n != -1:
            y.append(test_data[n][1])
            find = True
    if find is False:
        y.append(func3(i))
        if y[-1] > 95:
            y[-1] -= 0.013
        elif y[-1] > 90:
            y[-1] -= 0.018
        elif y[-1] > 85:
            y[-1] -= 0.004
        elif y[-1] > 80:
            y[-1] -= 0.001
        elif y[-1] > 75:
            y[-1] += 0.002
        elif y[-1] > 70:
            y[-1] += 0.0015
```



[2023-10-24 23:59:36]

[SECURITY] Checking security issues has passed!

[PROBLEM_01] Testing observed data

Evaluation metric: 1.486391

[SECURITY] Checking security issues has passed!

[PROBLEM_02] Testing unseen data

Evaluation metric: 4.462163

- Metric: 5.94855400
- Rank: 6
- Score: 97.5