Homework 1: Model fitting

By Romrawin Chumpu 639506093

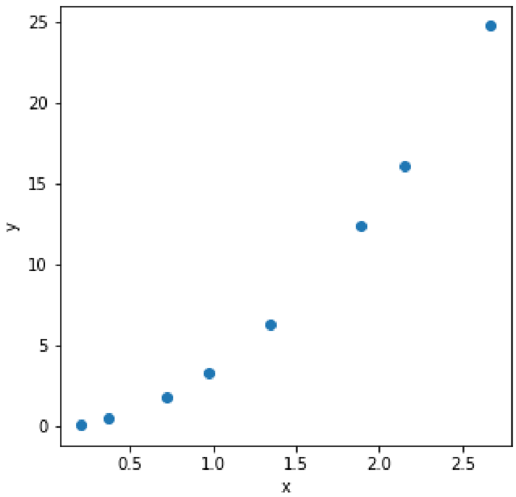
จากข้อมูลในตารางที่กำหนดให้ต่อไปนี้

x	y
0.2	0.02
0.37	0.48
0.72	1.80
0.97	3.26
1.34	6.23
1.89	12.40
2.15	16.04
2.67	24.74

จงหาฟังก์ชัน(แบบจำลอง)ที่มีความสอดคล้อง(สัมพันธ์)กับข้อมูลที่กำหนดให้มากที่สุดเท่าที่จะสามารถหาได้ พร้อมทั้งหาค่า R^2

Plot a graph to see the trend of given **x** and **y** (using python)

```
x = np.array([0.2, 0.37, 0.72, 0.97, 1.34, 1.89, 2.15, 2.67])
y = np.array([0.02, 0.48, 1.80, 3.26, 6.23, 12.40, 16.04, 24.74])
```



Set the hypotheses models from observed data

Because the trend are more curvature-like, the hypotheses models are intuitively narrowed down to these following models;

- A power curve
- An exponential curve
- A polynomial model

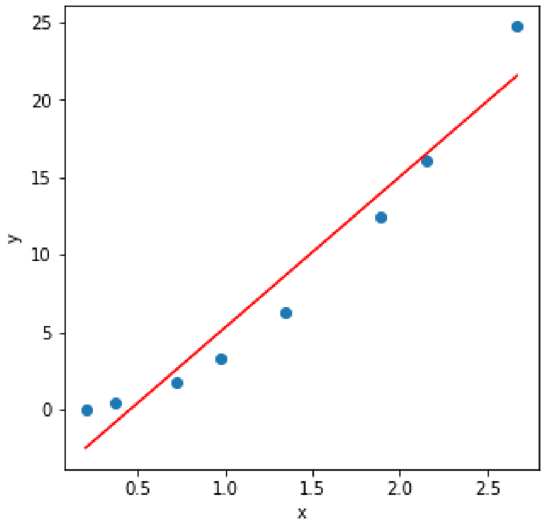
Fitting straight line (experiment for comparison)

Model: $y = ax + b$ from the data

```
# Find a
m = len(x)
sum_yx = np.sum(x*y)
sum_x_sum_y = np.sum(x)*np.sum(y)
sum_xx = np.sum(x**2)
sum_x2 = np.sum(x)**2

# Find b
sum_x2y = np.sum(x**2)*np.sum(y)
sum_x_sum_yx = np.sum(x)*np.sum(y*x)

a = ((m*sum_yx)-sum_x_sum_y)/((m*sum_xx)-sum_x2)
b = ((sum_x2y)-sum_x_sum_yx)/((m*sum_xx)-sum_x2)

print("a = %0.3f, and b = %0.3f" %(a, b))
```

a = 9.735, and b = -4.425

Final model: $y = 9.735x - 4.425$



Find R-squared

Sum-squared

```
SSE        30.114401
SSR       518.268486
SST       548.382887
```

R-squared = 0.945

	x	y	y_pred	SSE	SSR	SST
0	0.20	0.02	-2.478116	6.240585	112.346565	65.630252
1	0.37	0.48	-0.823106	1.698086	80.001507	58.388702
2	0.72	1.80	2.584268	0.615076	30.658174	39.958202
3	0.97	3.26	5.018106	3.090937	9.629502	23.631752
4	1.34	6.23	8.620187	5.712993	0.248938	3.576827
5	1.89	12.40	13.974631	2.479464	34.262073	18.307702
6	2.15	16.04	16.505823	0.216991	70.301069	62.706602
7	2.67	24.74	21.568207	10.060270	180.820658	276.182852

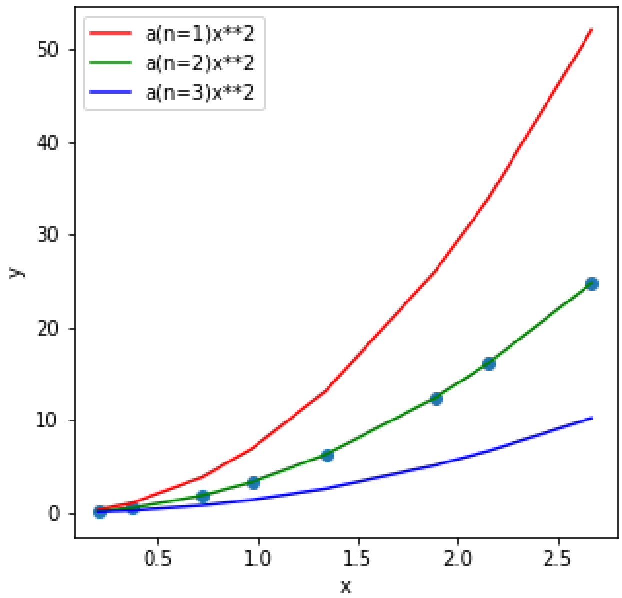
Fitting power curve

Model: $y = ax^n$

Find a by

```
def a_power(n):
  sum_yxn = np.sum(y*(x**n))
  sum_xn = np.sum(x**(2*n))
  a = sum_yxn/sum_xn
  return a
```

Compare y prediction with n=1, n=2, and n=3 power on a parameter



Select n = 2 as a candidate power curve fitting because the prediction results the closest to the actual y from this plot.

Final model: $y = 3.47x^2$

Find R-squared

Sum-squared

```
SSE        0.014187
SSR      546.476204
SST      548.382887
```

R-squared = 0.997

	x	y	y_pred	SSE	SSR	SST
0	0.20	0.02	0.138812	1.411629e-02	63.719317	65.630252
1	0.37	0.48	0.475084	2.416654e-05	58.463854	58.388702
2	0.72	1.80	1.799003	9.931079e-07	39.970801	39.958202
3	0.97	3.26	3.265205	2.709355e-05	23.581172	23.631752
4	1.34	6.23	6.231270	1.614029e-06	3.572023	3.576827
5	1.89	12.40	12.396258	1.400136e-05	18.275695	18.307702
6	2.15	16.04	16.041461	2.134941e-06	62.729745	62.706602
7	2.67	24.74	24.739421	3.355481e-07	276.163599	276.182852

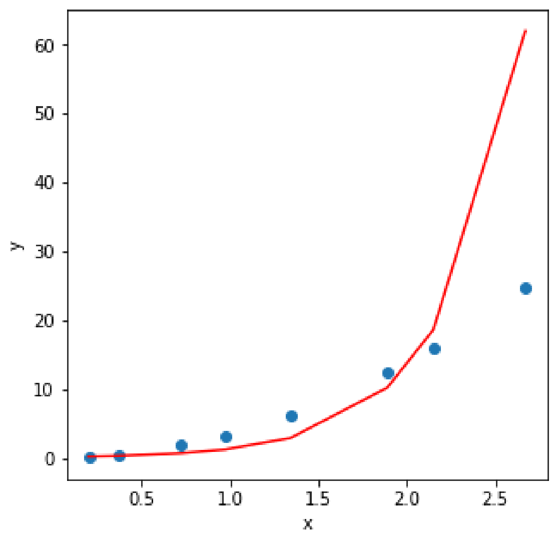
Fitting exponential curve

Model: $y = Ae^{Bx}$

```
# Find A
m = len(x)
sum_lnyx = np.sum(x*np.log(y))
sum_x_sum_lny = np.sum(x)*np.sum(np.log(y))
sum_xx = np.sum(x**2)
sum_x2 = np.sum(x)**2

# Find ln B
sum_x2lny = np.sum(x**2)*np.sum(np.log(y))
sum_x_sum_lnyx = np.sum(x)*np.sum(np.log(y)*x)

A = ((m*sum_lnyx)-sum_x_sum_lny)/((m*sum_xx)-sum_x2)
B = ((sum_x2lny)-sum_x_sum_lnyx)/((m*sum_xx)-sum_x2)

print("A = %0.3f, and B = %0.3f" %(A, np.e**B))
```

A = 2.314 and B = 0.129

Final model: $y = 0.129e^{2.314x}$



Find R-squared

Sum-squared

SSE	1418.391155
SSR	3274.597953
SST	548.382887

R-squared = 5.971

	x	y	y_pred	SSE	SSR	SST
0	0.20	0.02	0.204490	0.034037	62.675088	65.630252
1	0.37	0.48	0.303033	0.031317	61.124517	58.388702
2	0.72	1.80	0.681041	1.252068	55.356704	39.958202
3	0.97	3.26	1.214440	4.184315	47.704023	23.631752
4	1.34	6.23	2.858617	11.366226	27.695310	3.576827
5	1.89	12.40	10.204732	4.819202	4.340897	18.307702
6	2.15	16.04	18.623113	6.672474	110.289132	62.706602
7	2.67	24.74	62.023126	1390.031516	2905.412282	276.182852

Fitting polynomial Model

Model: $y = c_1x^n + c_2x^{n-1} + \dots + c_n$

Experiment polynomial models by order 2, 3, and 4

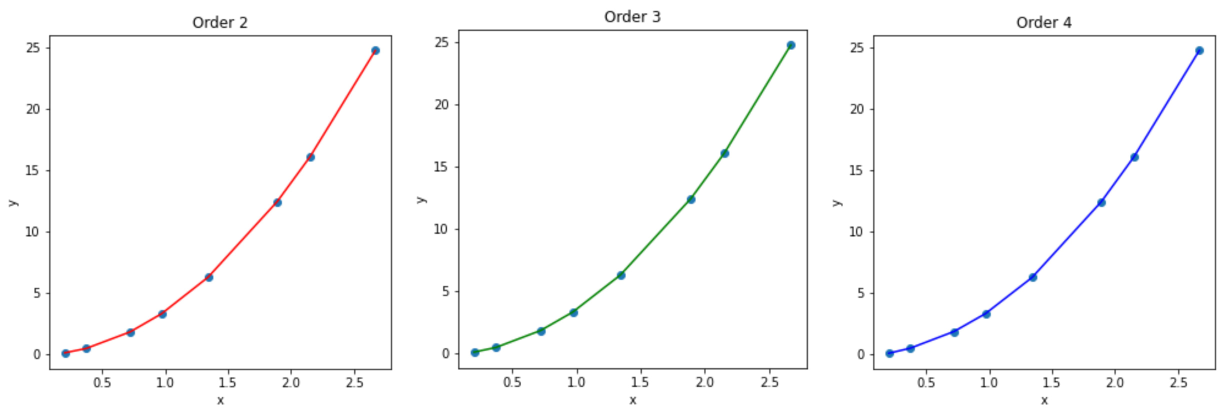
```
poly_model2 = np.poly1d(np.polyfit(x, y, 2))
print(np.polyfit(x, y, 2))
poly_model3 = np.poly1d(np.polyfit(x, y, 3))
print(np.polyfit(x, y, 3))
poly_model4 = np.poly1d(np.polyfit(x, y, 4))
print(np.polyfit(x, y, 4))
```

Final models:

Order 2: $y = 3.437x^2 + 0.116x - 0.086$

Order 3: $y = 0.041x^3 + 3.26x^2 + 0.319x - 0.139$

Order 4: $y = -0.082x^4 + 0.515x^3 + 2.372x^2 + 0.915x - 0.243$



Because the results are the same, then choose a polynomial order 2 model as the candidate.

Find R-squared

Sum-squared

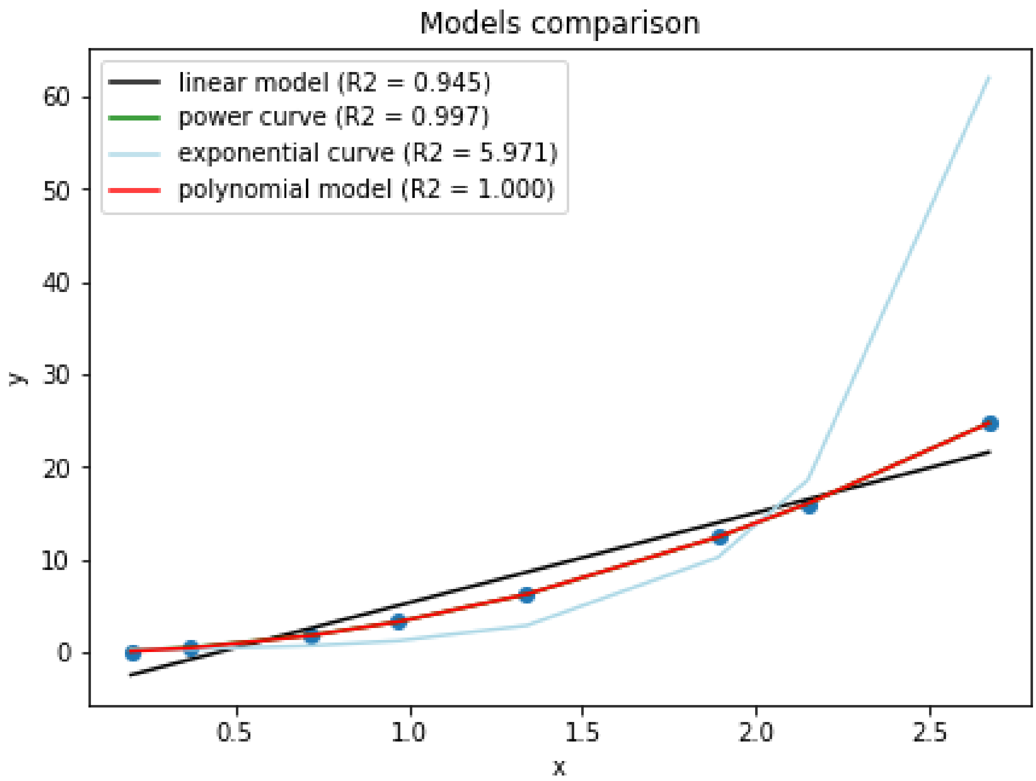
```
SSE          0.006733
SSR        548.378228
SST        548.382887
```

R-squared = 1.000

	x	y	y_pred	SSE	SSR	SST
0	0.20	0.02	0.074321	2.950801e-03	65.047925	65.630252
1	0.37	0.48	0.427067	2.801934e-03	59.185592	58.388702
2	0.72	1.80	1.778895	4.454286e-04	39.994353	39.958202
3	0.97	3.26	3.260080	6.471992e-09	23.467058	23.631752
4	1.34	6.23	6.240750	1.155721e-04	3.524718	3.576827
5	1.89	12.40	12.410747	1.154967e-04	18.235910	18.307702
6	2.15	16.04	16.051367	1.292016e-04	62.567844	62.706602
7	2.67	24.74	24.726773	1.749619e-04	276.354828	276.182852

Summary model fitting

Here is the comparison plot of all the model fittings from the previous calculation.



The polynomial model is the best fit for this data (with power 2, 3, and 4). Because, when compared to other models, this model generates the most accurate predictions, with R-squared equal to 1.000.