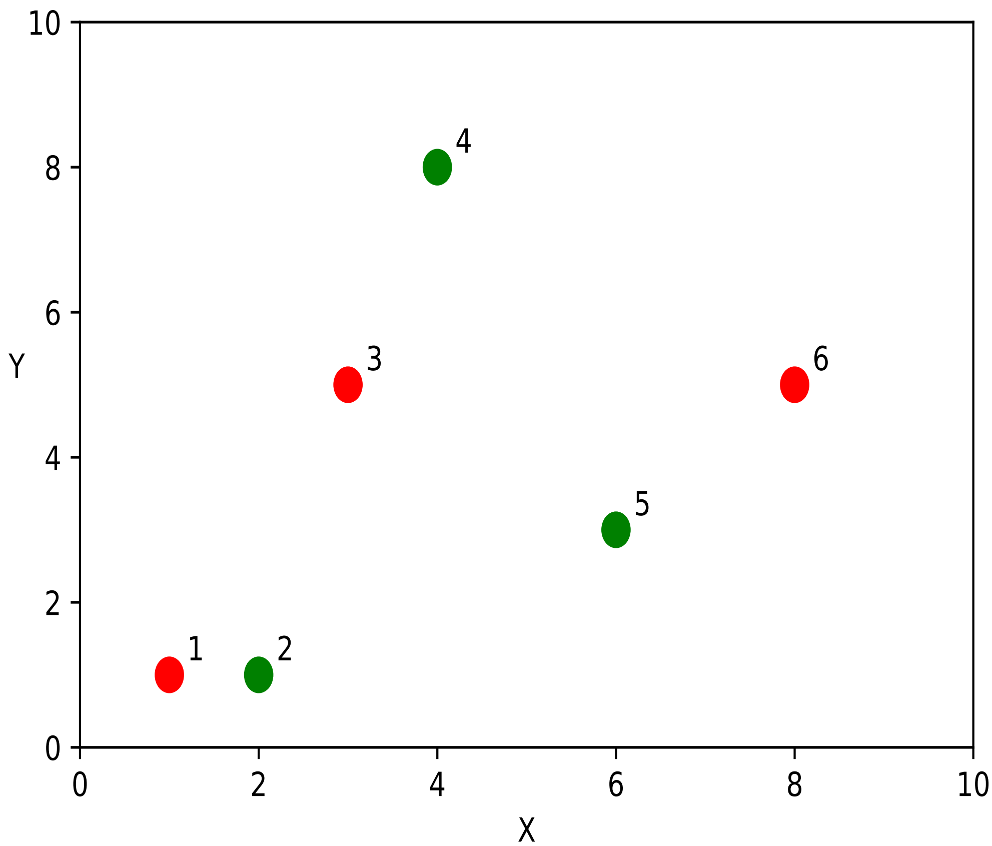
LINEAR

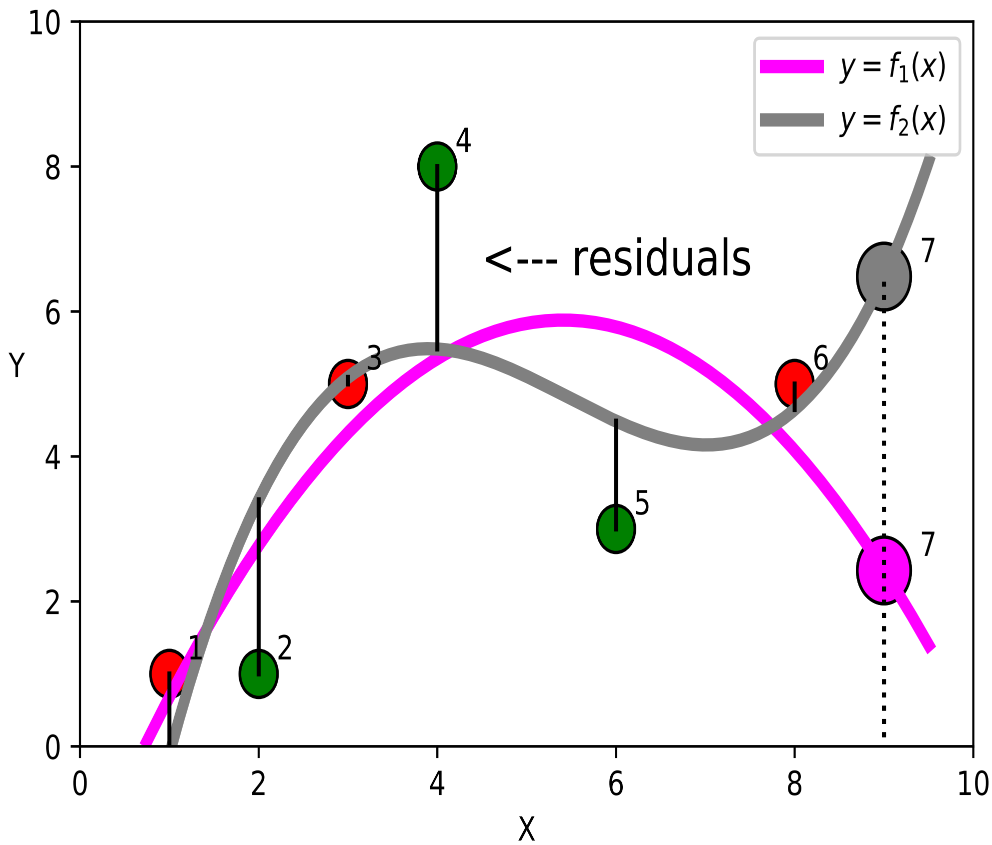
MODELS

Introduction



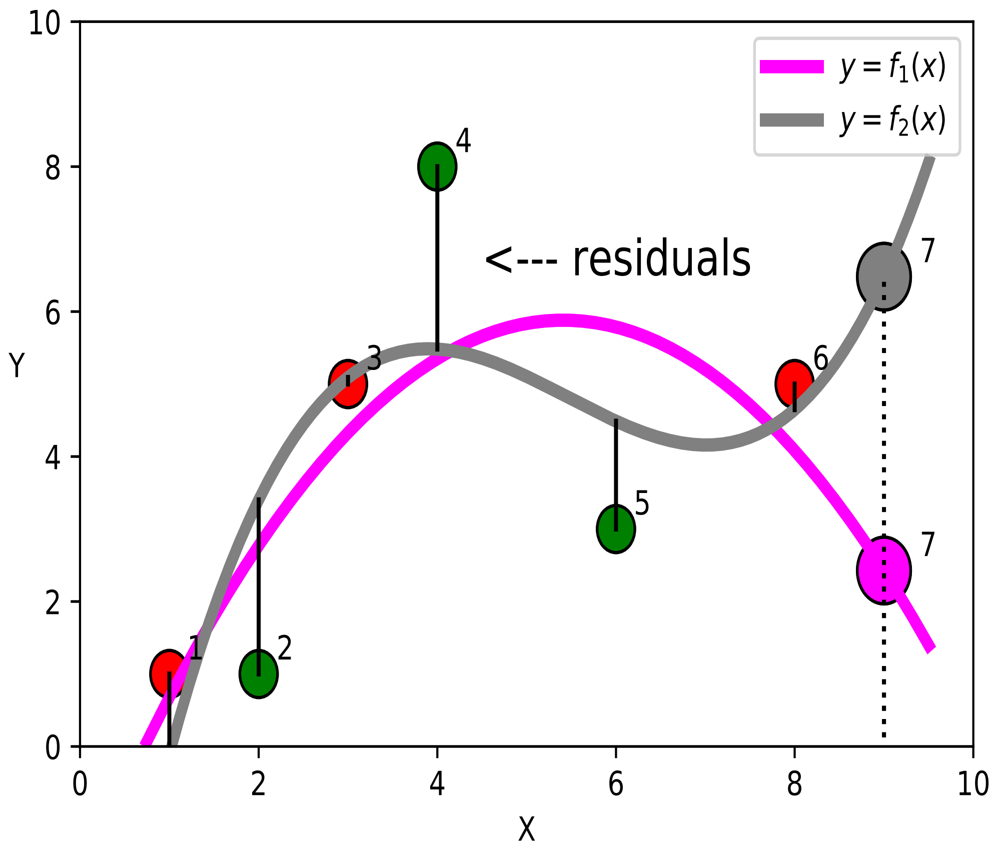
- n (explanatory) variables $x_1, \ldots, x_n$
- n response variables $y_1, \ldots, y_n$

Problem Statement



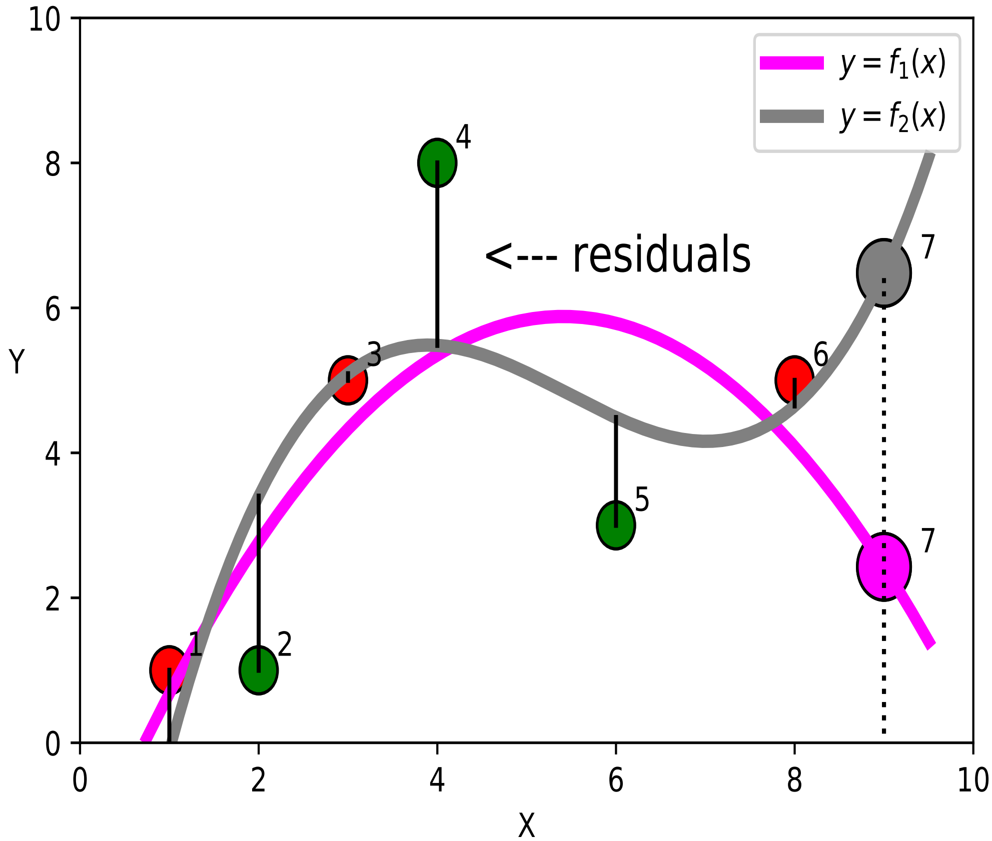
- find $f(x)$ to match data and give good prediction

How to choose y=f(x)?



- minimize errors $e_i = |f(x_i) - y_i|$

Error Criteria



- minimize errors

$$\text{Minimize:} \quad Err(x) = \frac{1}{2} e_i^2$$

Polynomial Functions

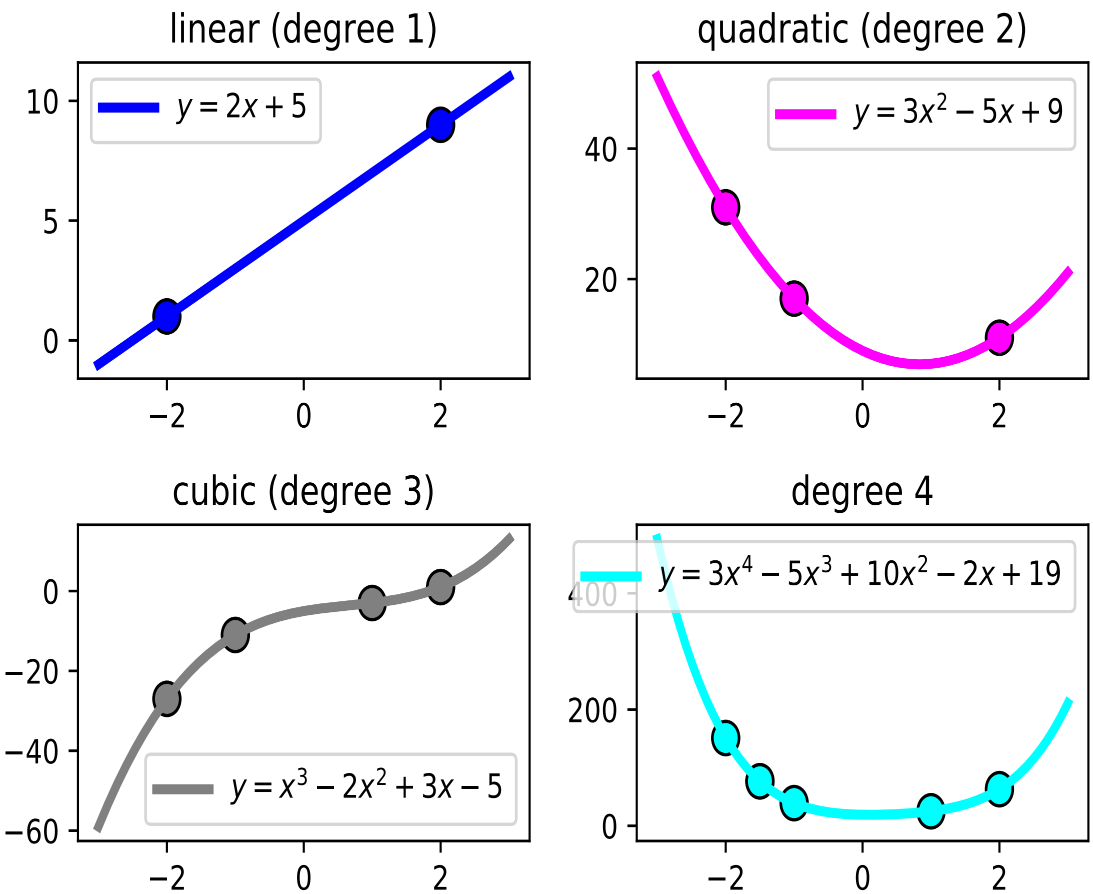
$$f(x, W) = w_n x^n + w_{n-1} x^{n-1} + \cdots + w_1 x + w_0$$

- n is the degree
- $W = (w_n, w_{n-1}, \ldots, w_1, w_0)$ weights
- linear in W:

  $f(x, W_1 + W_2) = f(x, W_1) + f(x, W_2)$
- not linear in x (for $n > 1$):

  $f(x + z, W) \neq f(x, W) + f(z, W)$
- linear models vs. linear functions

Example: Polynomials



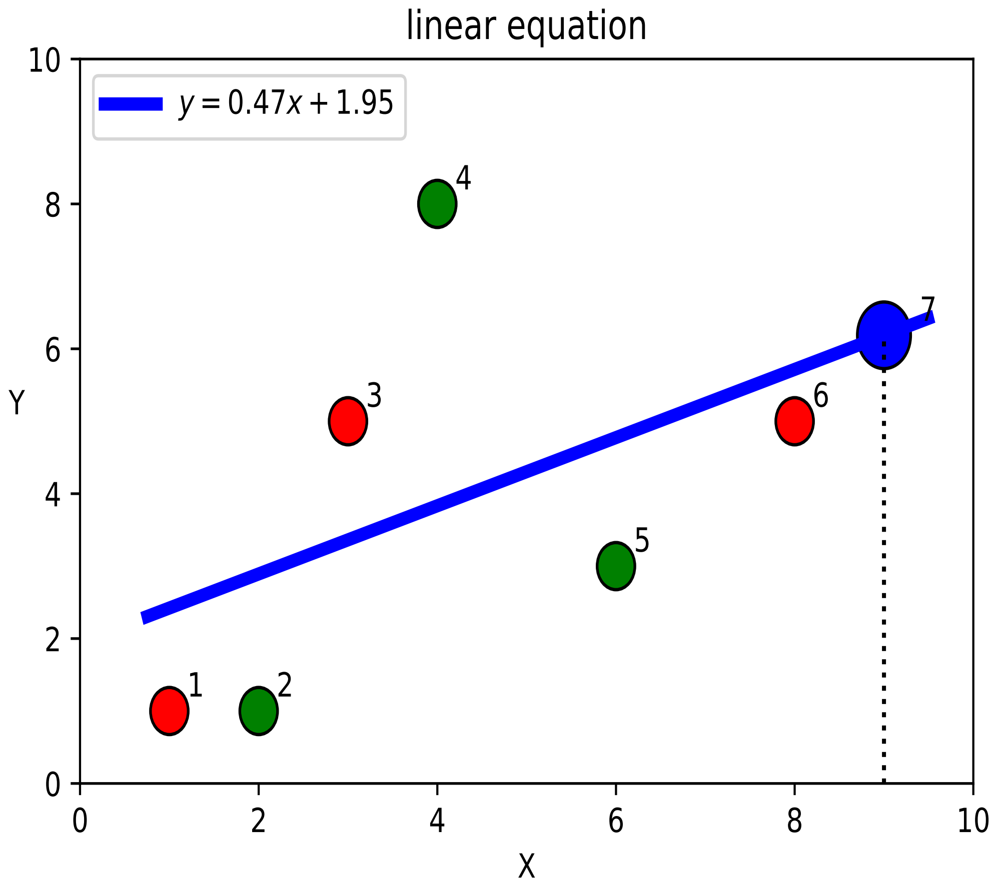
- how to address underfitting?
- increase model complexity

Python Code

```python
import numpy as np
from sklearn.metrics import mean_squared_error , r2_score

x = np.array([1 , 2, 3, 4, 6, 8])
y = np.array([1 , 1, 5, 8, 3, 5])
degree = 1
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

```
> weights

weights:  [ 0.47058824  1.95098039]

> model(9)

6.18627

> rmse

2.20997441798

>r2

0.204418418951
```

Linear Polynomial

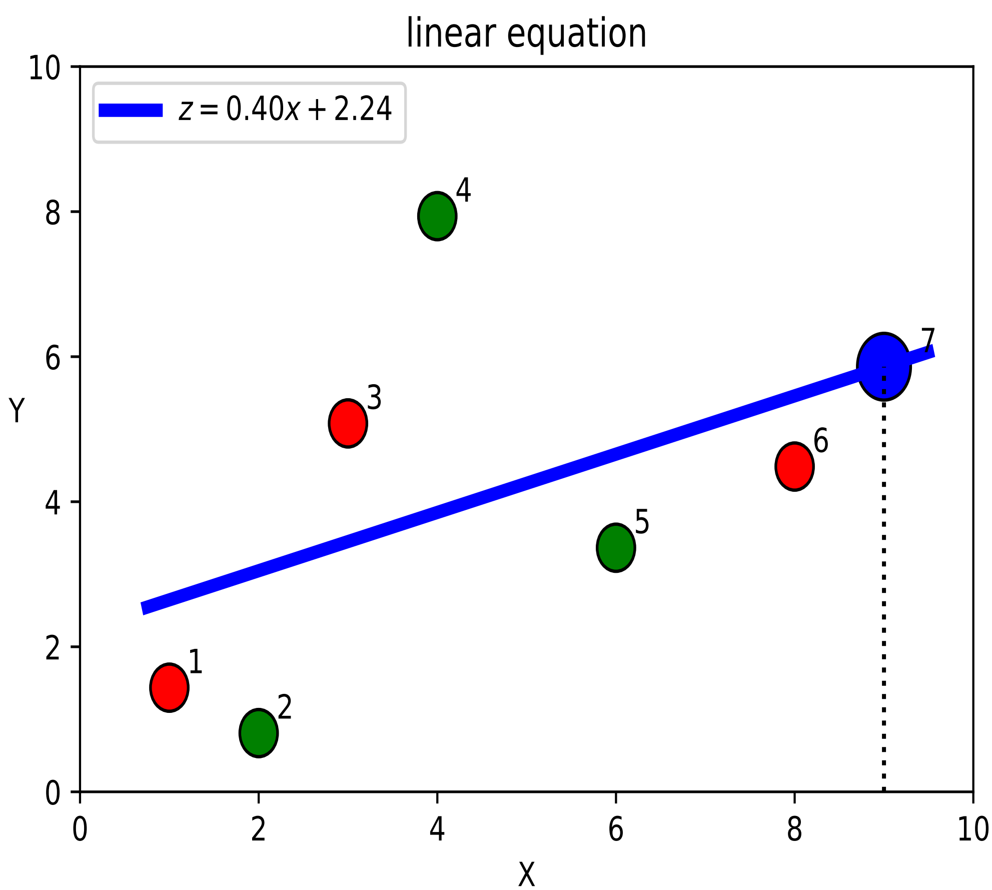


$$\text{rmse} = 2.21, r^2 = 0.20, y_7 = 6.18$$

Python Code

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
id_list = ['1','2','3','4','5','6']

degree = 1
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights)

x_points = np.linspace(0.75,9.5, 1000)
y_points = model(x_points)

ax, fig = plt.subplots()
plt.xlim(0, 10)
plt.ylim(0, 10)
plt.xlabel('X')
plt.ylabel('Y', rotation=0)
plt.plot(x_points, y_points, lw=4, color='blue')
plt.scatter(x, y, color='black', s=200)
for i, txt in enumerate(id_list):
    plt.text(x[i]+0.2, y[i]+0.2, txt, fontsize=10)

x_new = 9
y_new = model(x_new)
plt.scatter(x_new, y_new, color='blue', edgecolor='k', s=400)
plt.plot([x_new, x_new],[0, y_new], color='black', ls='dotted')
plt.text(x_new +0.4, y_new +0.2, '7', fontsize=10)
plt.show()
```

Linear Polynomial with Noise



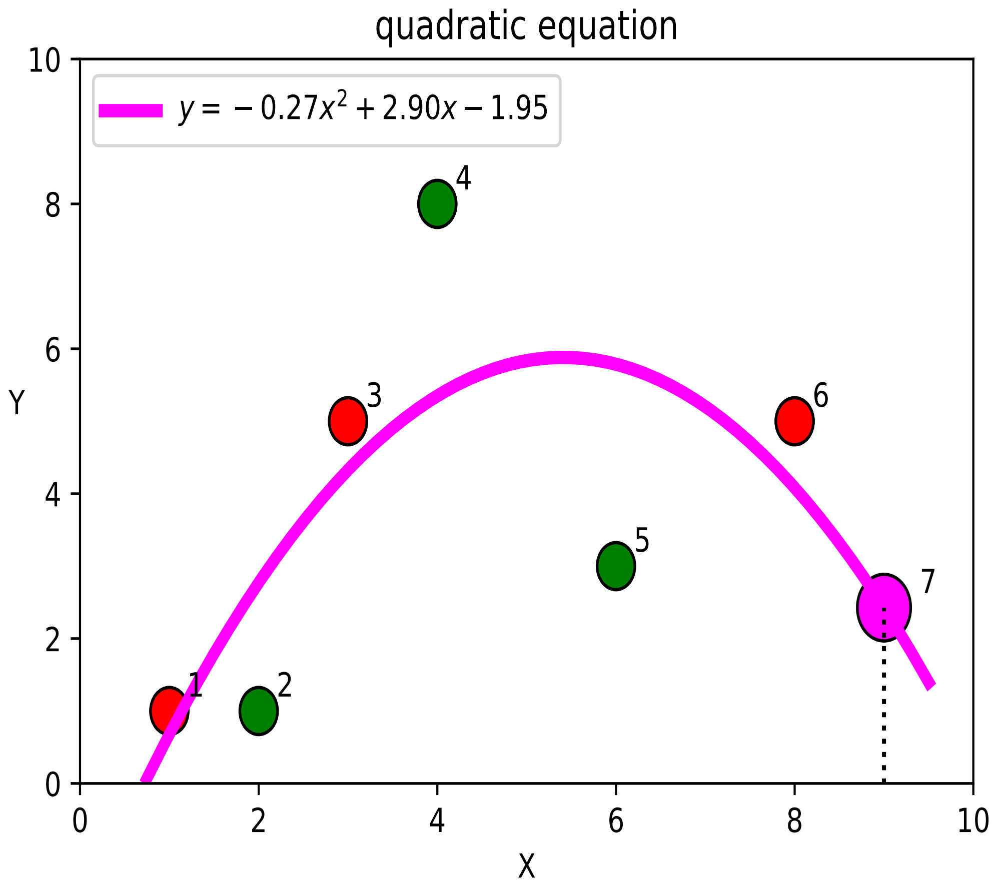
$$\text{rmse} = 2.18, r^2 = 0.16, y_7 = 5.86$$

Python Code

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
noise = np.random.normal(loc=0,scale=0.1,size=(6,))
y = y + noise

weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

```
> weights
weights:   [ 0.4018573   2.24487251]
> model(9)
5.86158818758
> rmse
2.17709017809
>r2
0.161827384707
```

Quadratic Polynomial



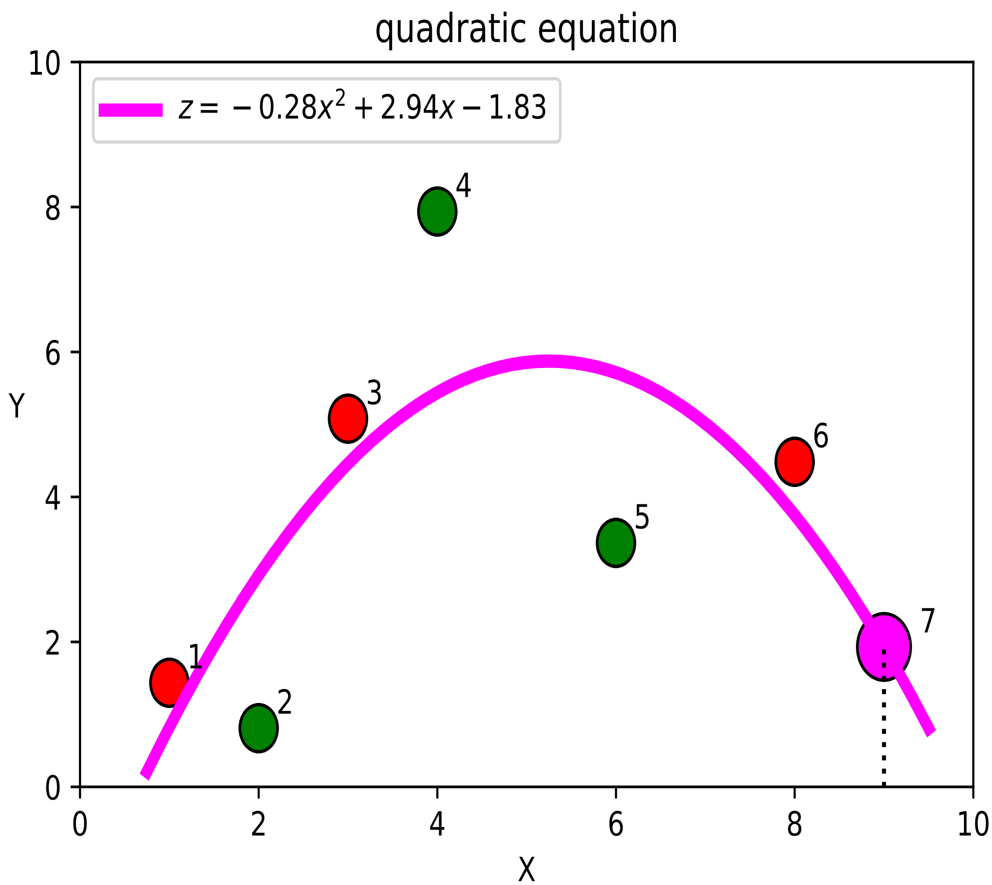
$$rmse = 1.79, r^2 = 0.48, y_7 = 2.43$$

Python Code

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
degree = 2
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

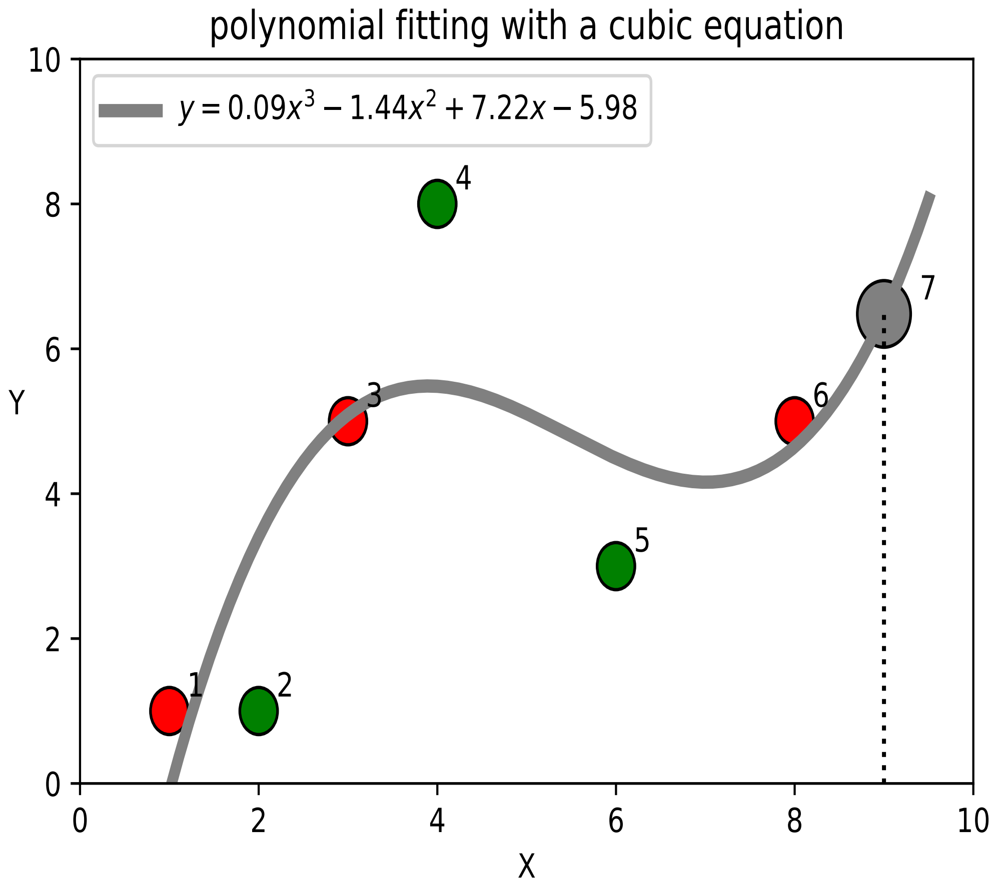
```
> weights
[-0.26774648  2.89605634 -1.94971831]
> model(9)
2.42732394366
> rmse
1.79461243805
>r2
0.475371869224
```

Quadratic Polynomial with Noise



$$rmse = 1.71, r^2 = 0.48, y_7 = 1.93$$

Cubic Polynomial



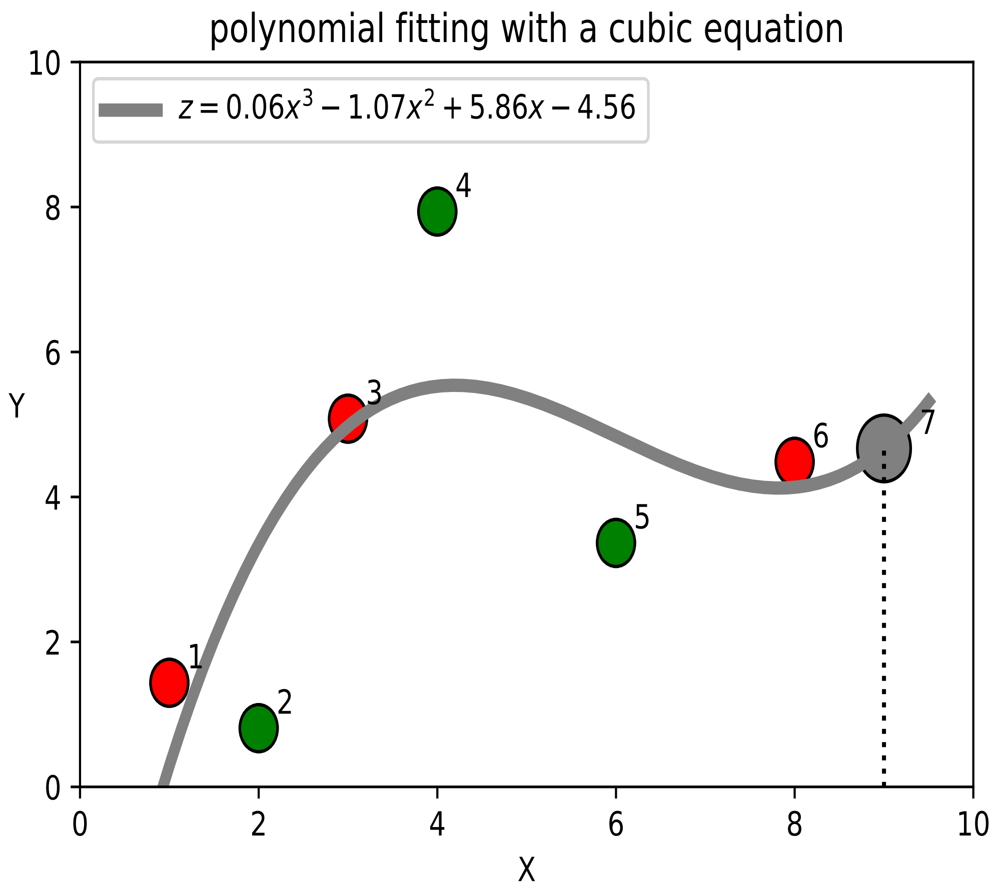
$$rmse = 1.62, r^2 = 0.58, y_7 = 6.48$$

Python Code

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
degree = 3
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

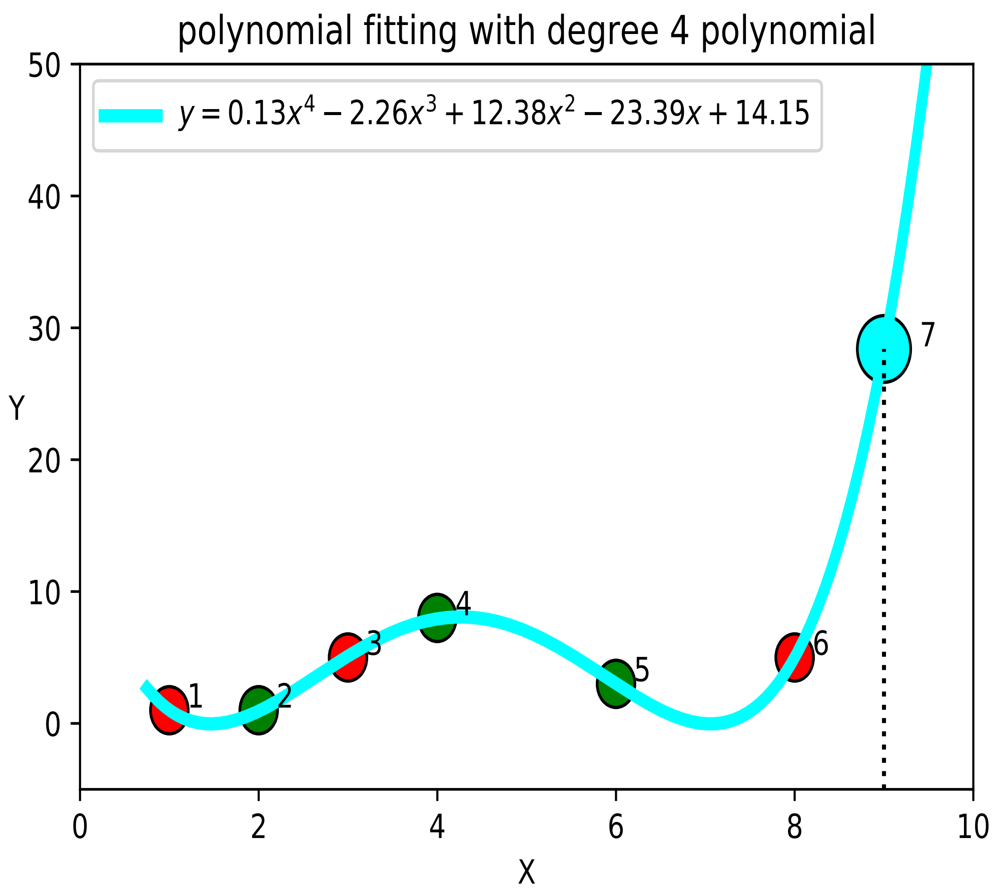
```
> weights
[ 0.08811052 -1.44167642  7.22280401 -5.9
> model(9)
6.4812020294
> rmse
 1.61723561985
>r2
0.573953675095
```

Cubic Polynomial with Noise



$$rmse = 1.62, r^2 = 0.53, y_7 = 4.67$$

Degree 4 Polynomial



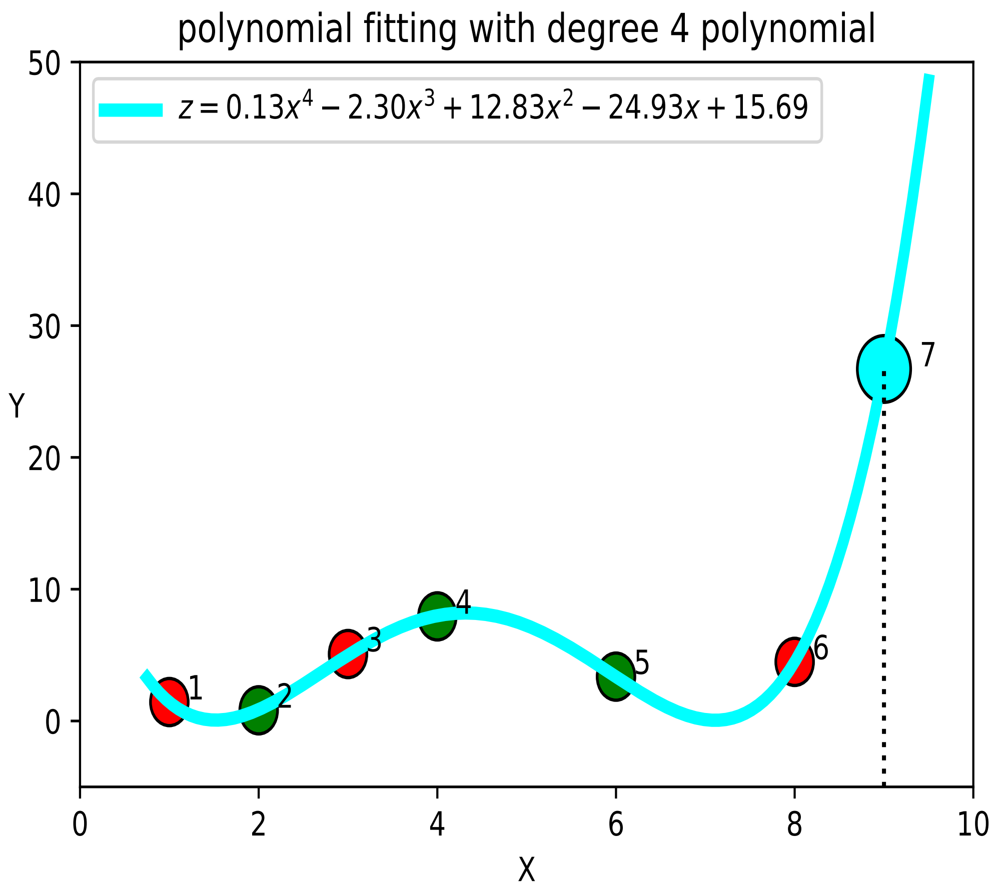
$$rmse = 0.06, r^2 = 0.99, y_7 = 28.40$$

Python Code

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
degree = 4
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

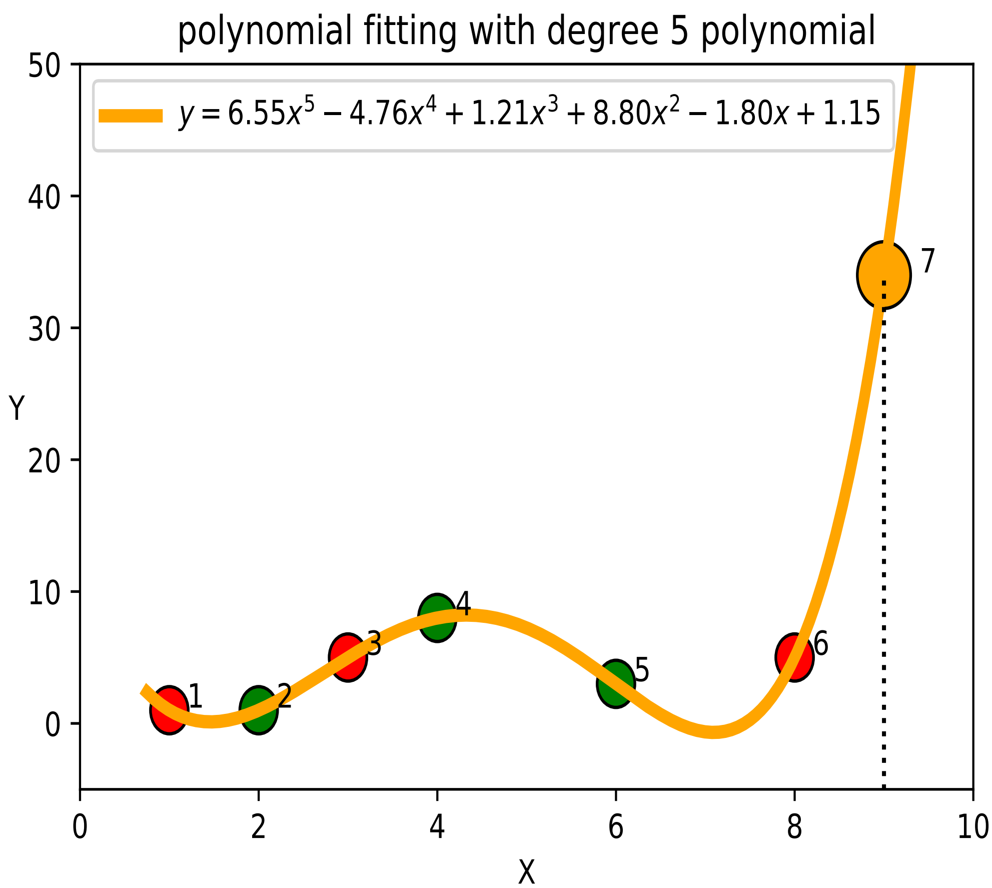
```
> weights
[  0.13271053  -2.26207602  12.38399123
 -23.39108187  14.15189474]
> model(9)
28.3957894737
> rmse
0.0594811877479
>r2
0.999423672303
```

Degree 4 Polynomial with Noise



$$rmse = 0.05, r^2 = 0.99, y_7 = 26.71$$

Degree 5 Polynomial



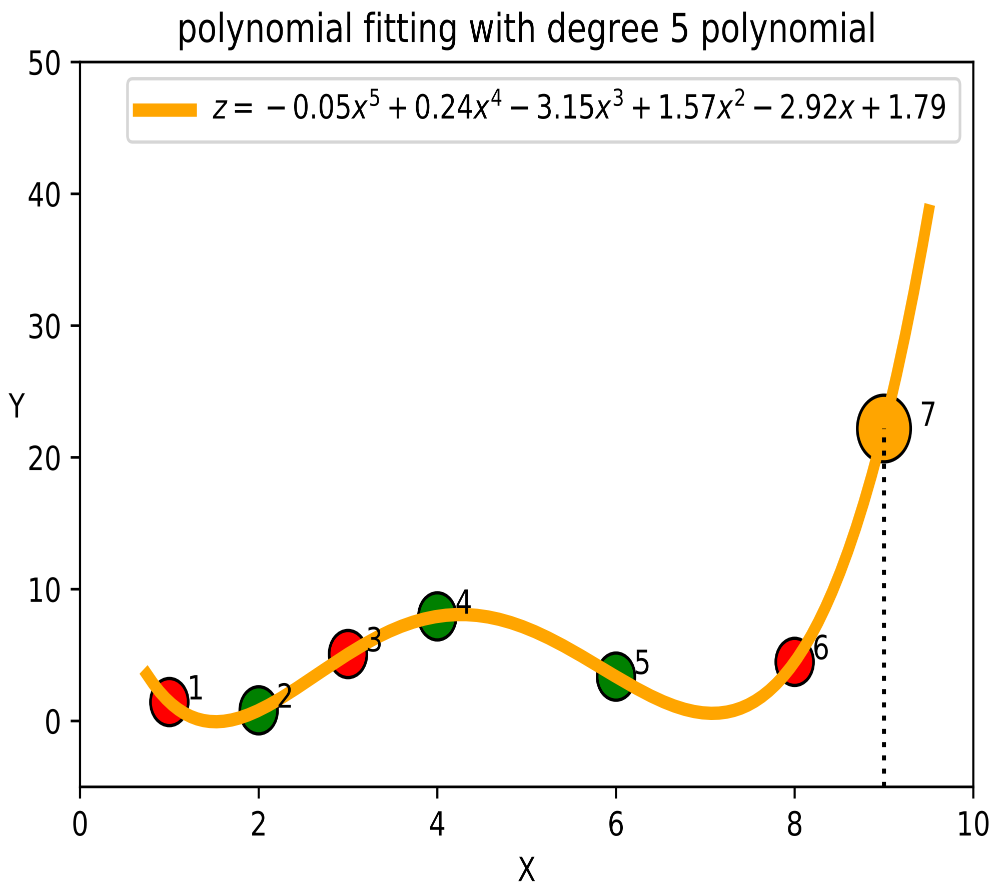
$$rmse = 0.06, r^2 = 0.99, y_7 = 34.0$$

Python Code

```
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 6, 8])
y = np.array([1, 1, 5, 8, 3, 5])
degree = 5
weights = np.polyfit(x,y, degree)
model = np.poly1d(weights_1)
predicted = model(x)
rmse = np.sqrt(mean_squared_error(y,predicted))
r2 = r2_score(y,predicted)
```

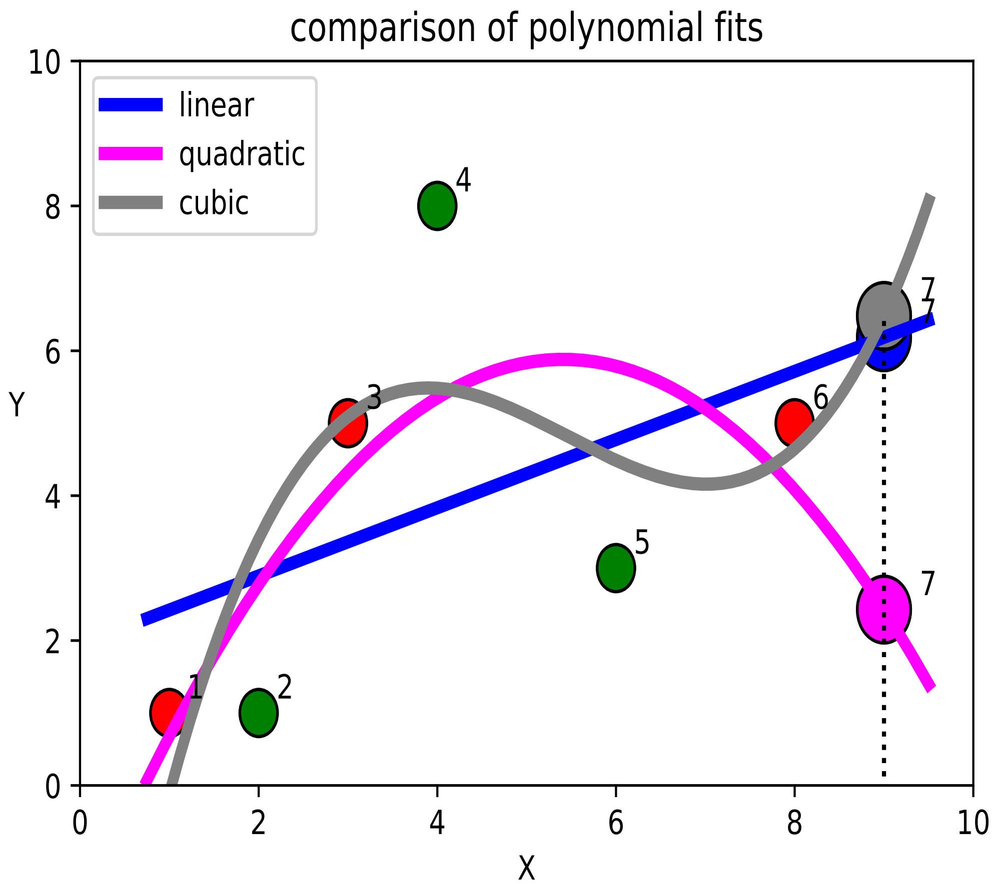
```
> weights
[  6.54761905e-03  -4.76190476e-03  -1.21
 8.79761905e+00   -1.80452381e+01   1.145
> model(9)
34.0
> rmse
0.0594811877479
>r2
0.999423672303
```

Degree 5 Polynomial with Noise

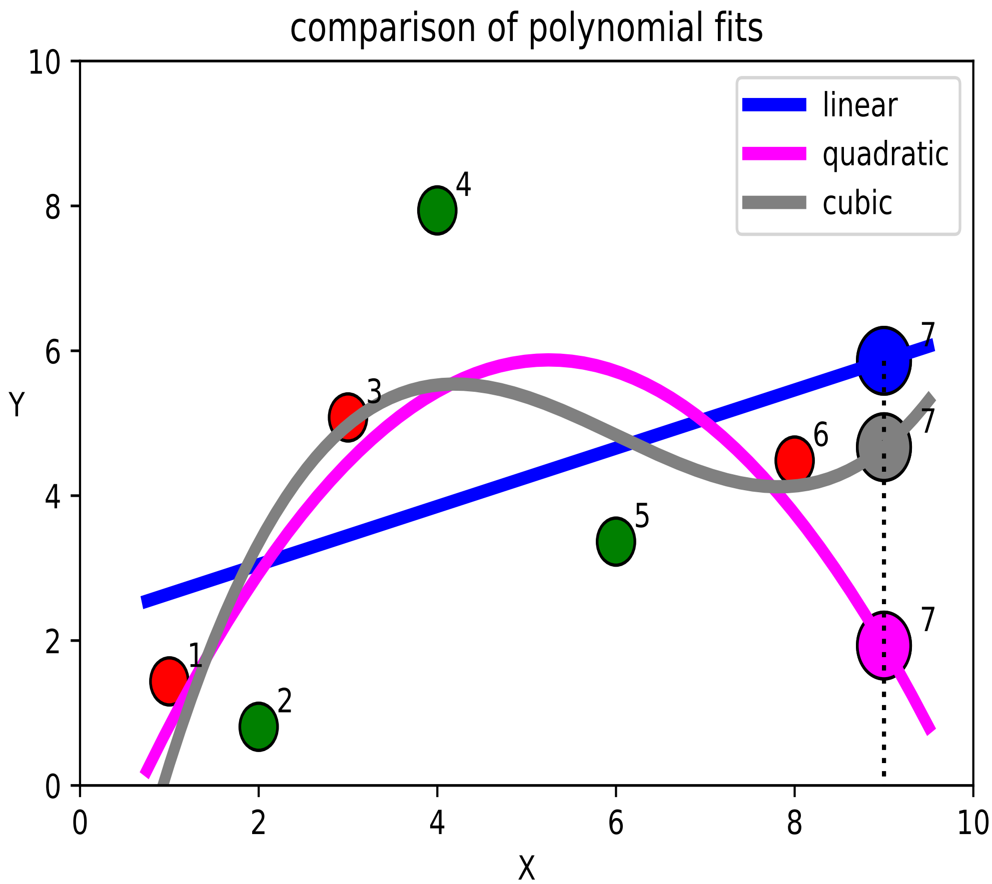


$$rmse = 0.05, r^2 = 0.99, y_7 = 22$$

Comparing Polynomial Fitting



Comparing Polynomial Fitting with Noise



Comparing Polynomial Fitting

degree	RMSE	r^2	prediction $f(x_7)$
1	2.21	0.20	6.18
2	1.79	0.48	2.43
3	1.62	0.58	6.48
4	0.06	0.99	28.40
5	0.06	0.99	34.0

- hor higher degree, we capture existing data
- but unable to generalize well
- bias vs. variance trade-off

Effect of Noise

degree	original data: $y(x)$	$y(x_7)$
1	$0.47x + 1.95$	6.18
2	$-0.27x^2 + 2.90x - 1.95$	2.43
3	$0.09x^3 - 1.44x^2 + 7.22x - 5.98$	6.48
4	$0.13x^4 - 2.26x^3 + 12.38x^2 - 23.39x + 14.15$	28.40
5	$6.55x^5 - 4.76x^4 + 1.21x^3 + 8.80x^2 - 1.80x + 1.15$	34.0
degree	Noisy data: $z(x)$	$z(x_7)$
1	$0.40x + 2.24$	5.86
2	$-0.28x^2 + 2.94x - 1.83$	1.93
3	$0.06x^3 - 1.07x^2 + 5.86x - 4.56$	4.67
4	$0.13x^4 - 2.30x^3 + 12.83x^2 - 24.93x + 15.69$	26.71
5	$-0.05x^5 + 0.24x^4 - 3.15x^3 + 1.57x^2 - 2.92x + 1.79$	22.20

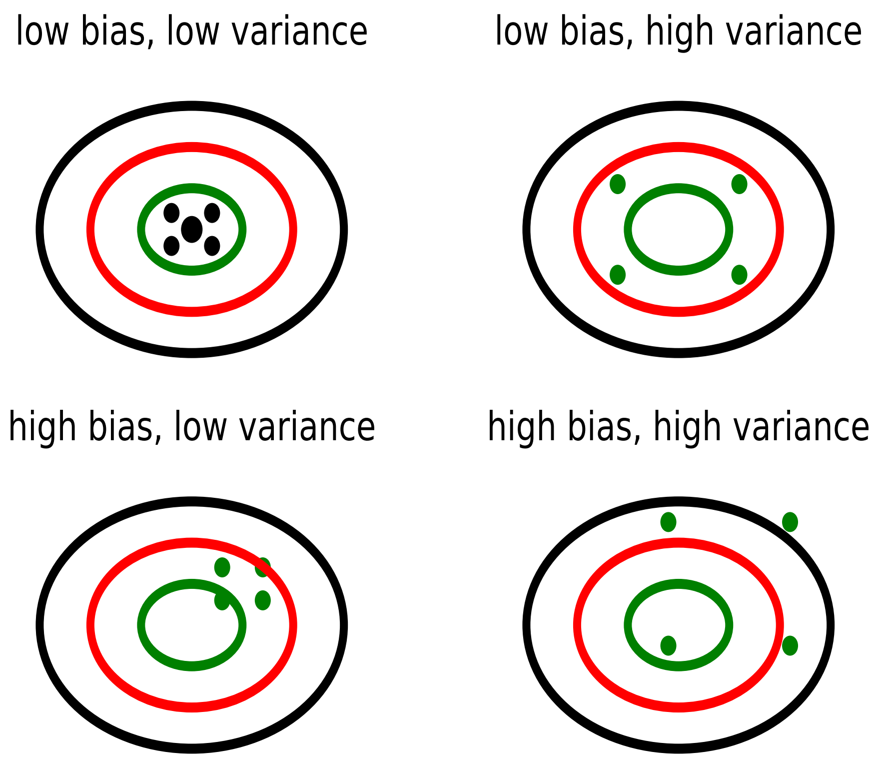
- small variations could result in large changes in high-degree coefficients
- difficult to generalize

Bias vs. Variance

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{ Irreducible Error}$$

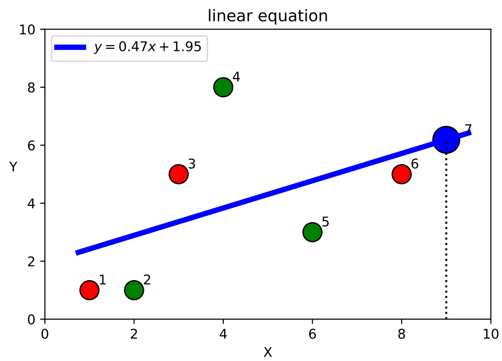
- bias: measures accuracy of average prediction
- variance: measures sensitivity to data set
- models trade bias and variance

Bias vs. Variance

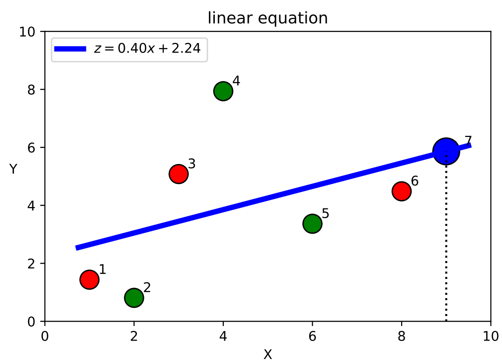


- ideal: low variance, low bias

Underfitting



(a) No Noise



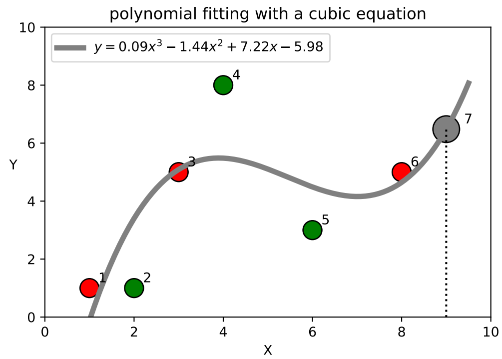
(b) With Noise

degree	original data: $y(x)$	$y(x_7)$
1	$0.47x + 1.95$	6.18

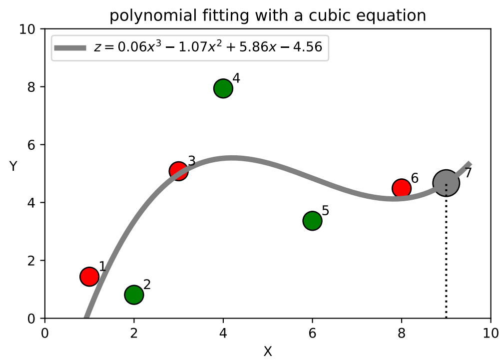
degree	Noisy data: $z(x)$	$z(x_7)$
1	$0.40x + 2.24$	5.86

- high bias, low variance

Bias vs. Variance



(a) No Noise



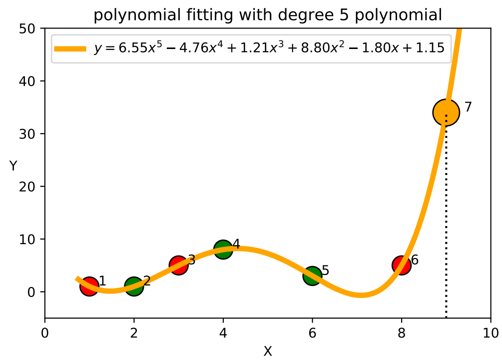
(b) With Noise

degree	original data: $y(x)$	$y(x_7)$
3	$0.09x^3 - 1.44x^2 + 7.22x - 5.98$	6.48

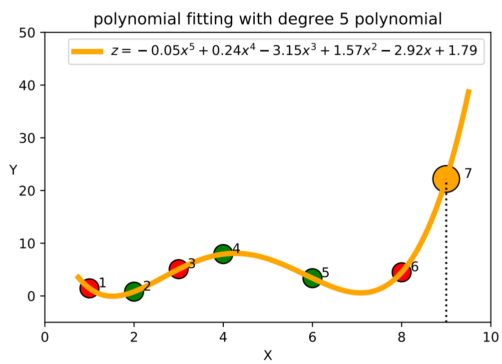
degree	Noisy data: $z(x)$	$z(x_7)$
3	$0.06x^3 - 1.07x^2 + 5.86x - 4.56$	4.67

- low bias, high variance

Overfitting



(a) No Noise



(b) With Noise

degree	original data: $y(x)$	$y(x_7)$
5	$6.55x^5 - 4.76x^4 + 1.21x^3 + 8.80x^2 - 1.80x + 1.15$	34.0

degree	Noisy data: $z(x)$	$z(x_7)$
5	$-0.05x^5 + 0.24x^4 - 3.15x^3 + 1.57x^2 - 2.92x + 1.79$	22.20

- high bias, low variance

How to Choose Polynomial Models

- want to avoid high degree polynomials

1. choose a class such as linear functions
2. use regularization to lower coefficients

$$Err(x) = \frac{1}{2}e_i^2 + \frac{\lambda}{2}|W|^2$$