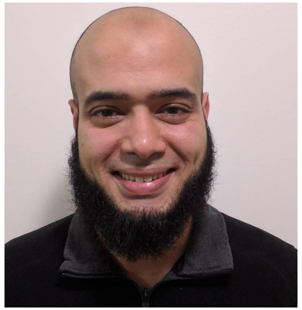
Machine Learning
Polynomial Regression

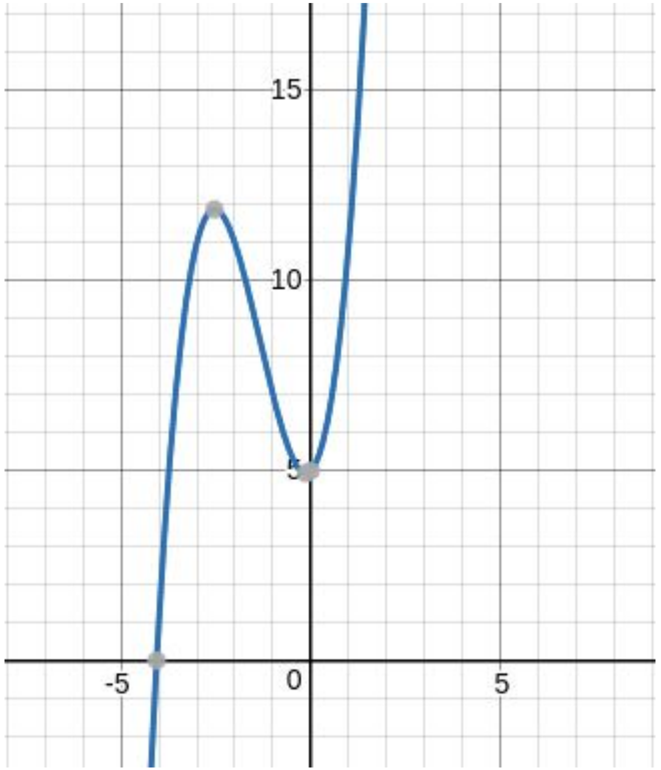
Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Background: Polynomial Function

- $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$
- If the max power of x is 0,
  this is a **constant** (y = 5)
- If the max power of x is 1,
  this is a **line** (y = 2x+3)
- If the max power of x is 2,
  this is a **quartic** function ($y = 2x+3x^2 -5$)
- This figure shows the cubic function: for: $y=x^3+4x^2+x+5$
  - Third order (largest exponent is 3)



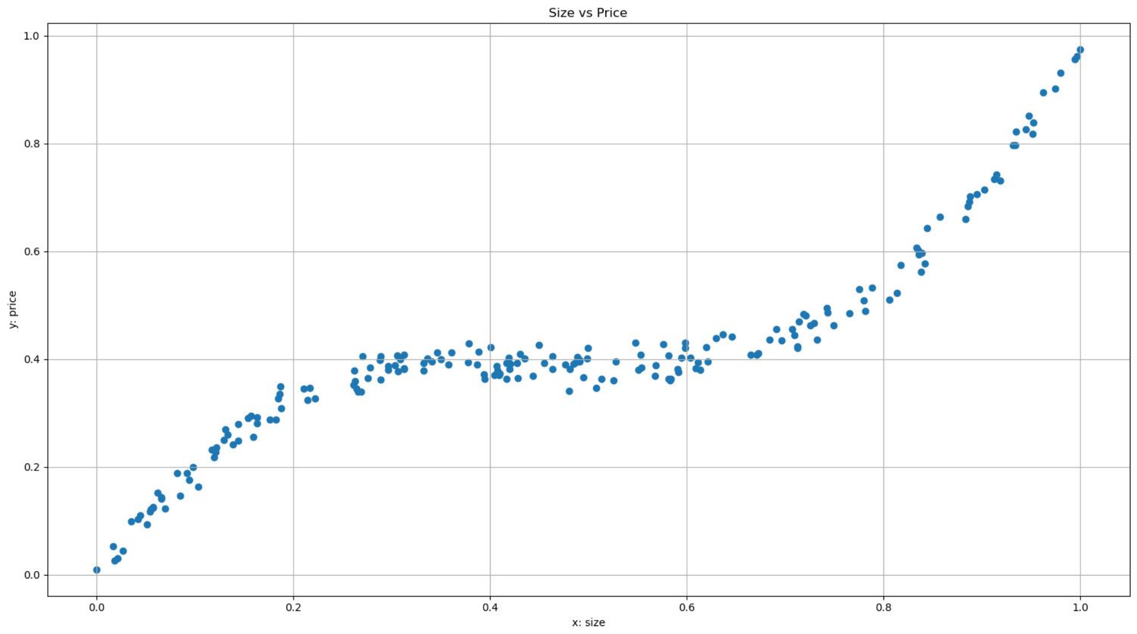
Background: Is this a Linear or Cubic Equation?

- **$y = mx^3 + c$**

- We can interpret this equation in 2 ways:
- We take m and c to be given values, and x the target parameter
  - Then this equation is **quadratic** in x
- Or m and c are the target parameters, and x^3 is just a given value
  - If so, this equation is a **linear** function of m and c
- In machine learning, x is usually just an input. m and c are the parameters
  - Then x^3 is just a value
  - Hence, in this context, the second case is true
  - This means this polynomial equation is just a linear equation for us!

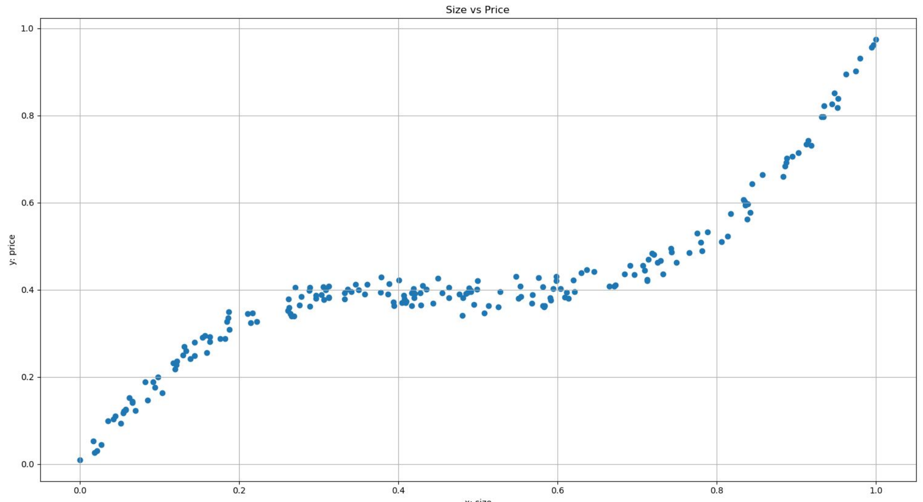
Can we fit a line for this data?

- Does this data satisfy the linear assumption?!



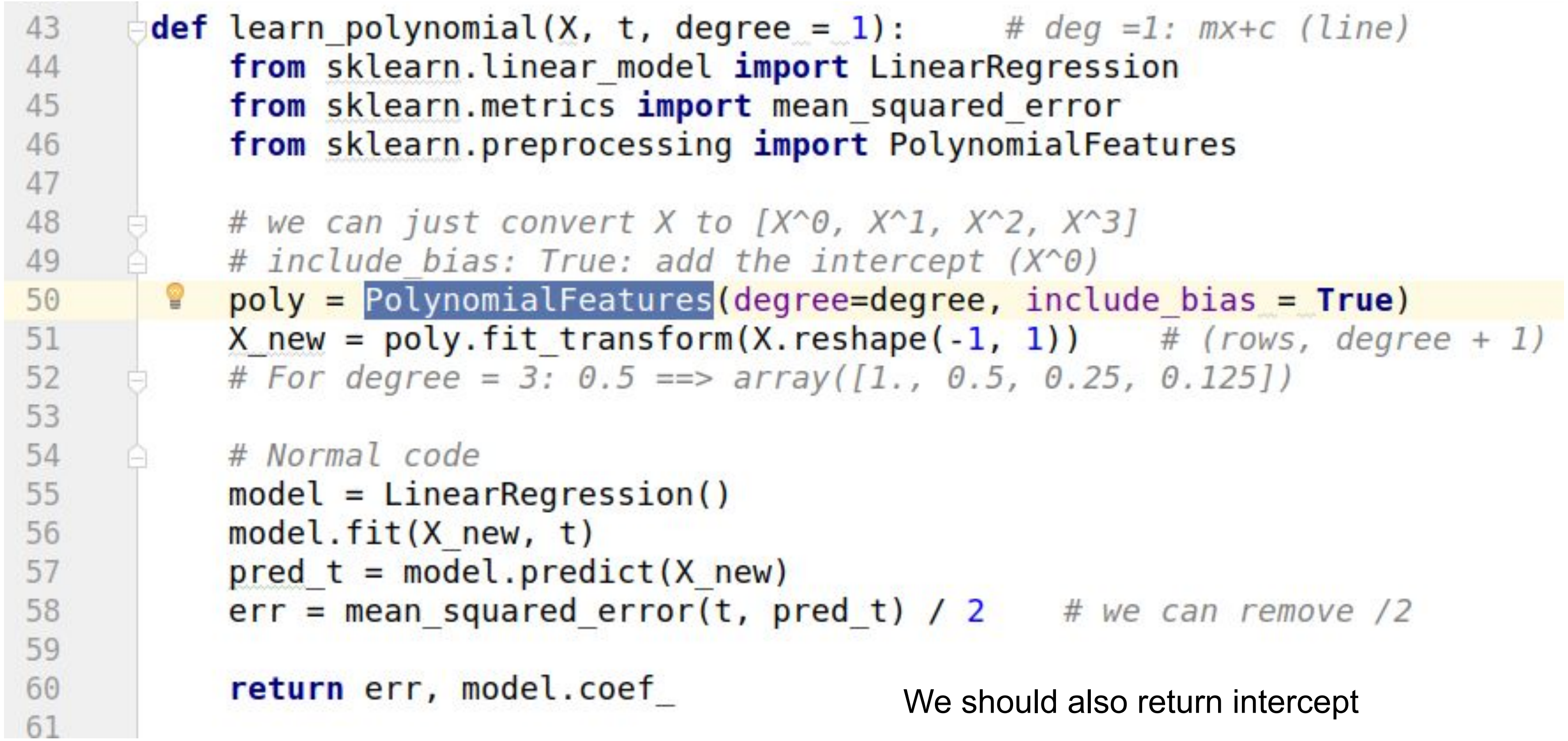
Fitting a Polynomial

- This data came from scaled: $\mathbf{y = 5 + x + 4x^2 + x^3 + noise}$
    - Clearly, **non-linear** curve(data)!
- Similar to **multivariate linear regression**, this can be written as:
  $f(x, W) = w_0 + w_1x + w_2x^2 + w_3x^3 + \text{noise}$
    - Where w0=5, w1=w3=1, w2=4
    - *The first 2 terms are like c+mx*
- This is a linear equation in W
    - Its polynomial features, x^2 and x^3, generate non-linear data!



How can we find W?

- We find the parameters W with the the **same code** of linear regression!
- To model: $f(x, W) = w_0 + w_1x + w_2x^2 + w_3x^3 + \text{noise}$
- Just think of this equation as having three input features:
    - $X1 = X$, $X2 = X^2$, $X3 = X^3$
    - To account for the intercept (bias): $X0 = 1$
- Now, apply the code to get the weights W as we did before!

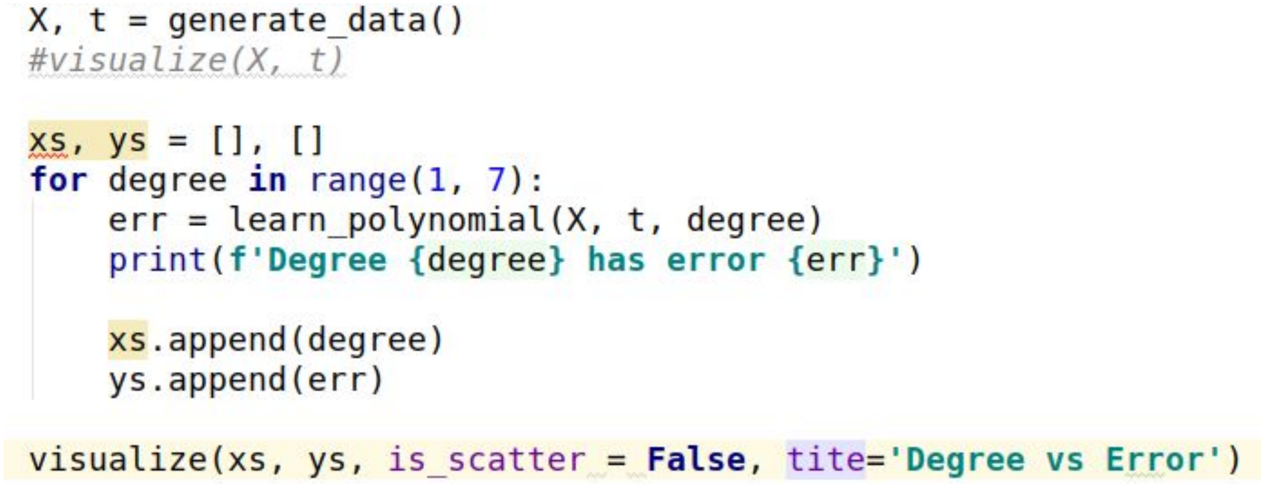


```python
def learn_polynomial(X, t, degree = 1):     # deg =1: mx+c (line)
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error
    from sklearn.preprocessing import PolynomialFeatures

    # we can just convert X to [X^0, X^1, X^2, X^3]
    # include_bias: True: add the intercept (X^0)
    poly = PolynomialFeatures(degree=degree, include_bias = True)
    X_new = poly.fit_transform(X.reshape(-1, 1))    # (rows, degree + 1)
    # For degree = 3: 0.5 ==> array([1., 0.5, 0.25, 0.125])

    # Normal code
    model = LinearRegression()
    model.fit(X_new, t)
    pred_t = model.predict(X_new)
    err = mean_squared_error(t, pred_t) / 2    # we can remove /2

    return err, model.coef_
```

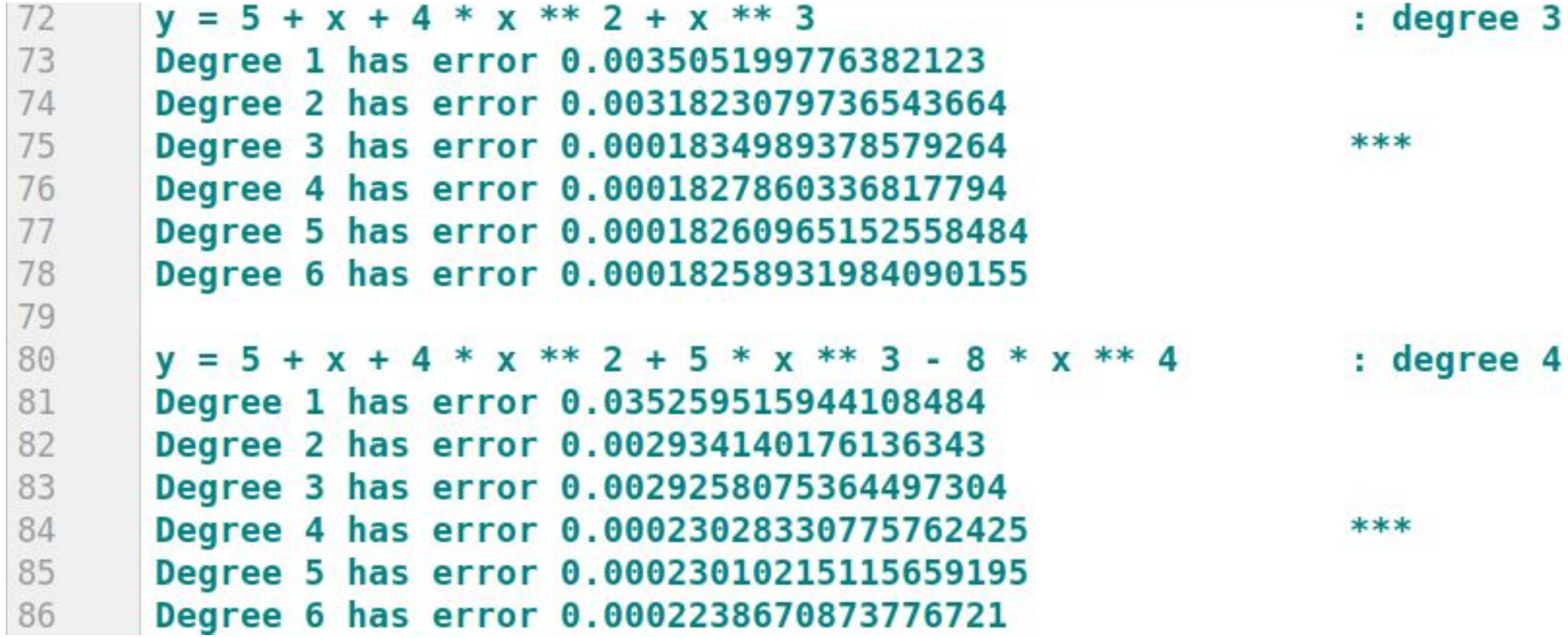
We should also return intercept

Learning Several Models!

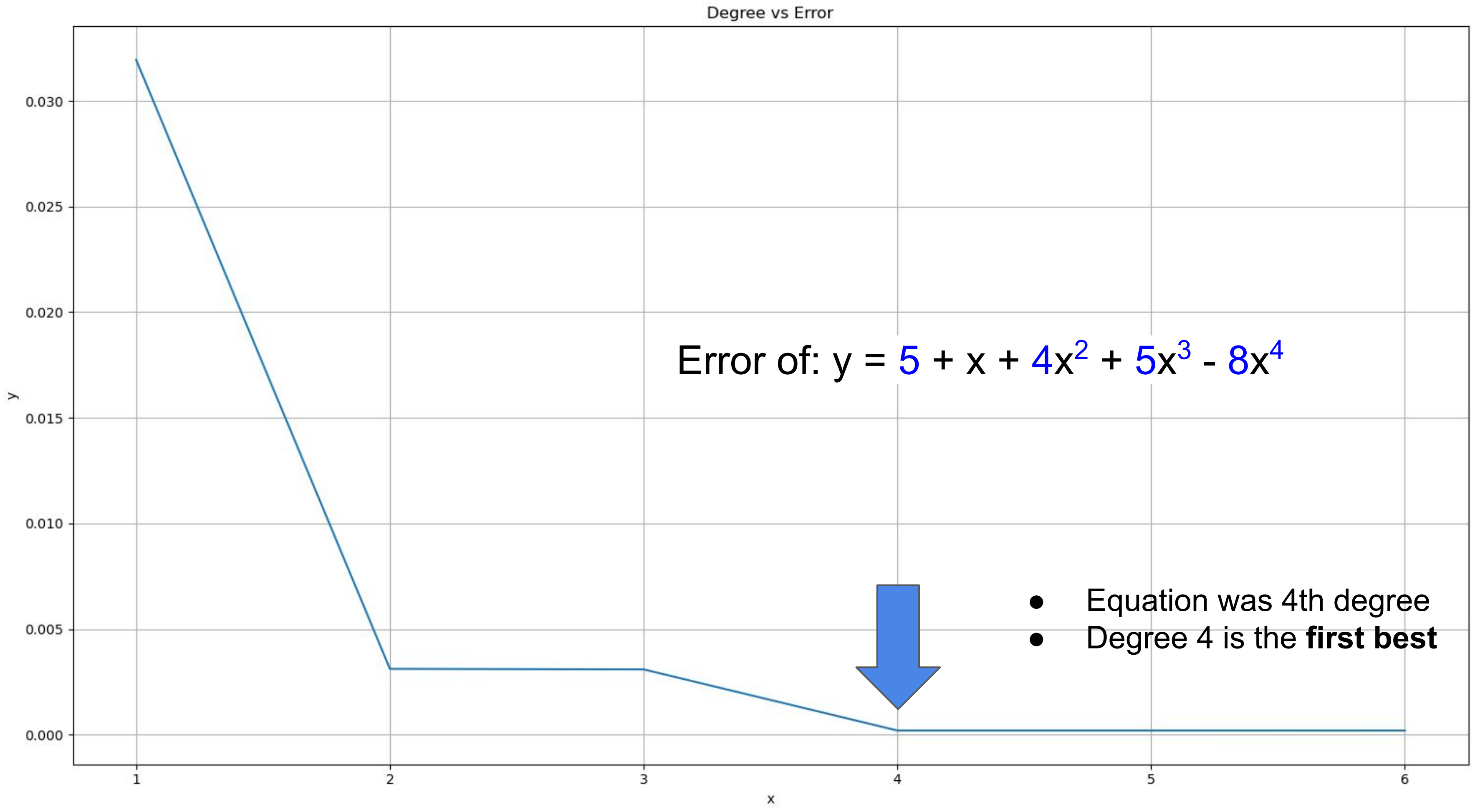
- Observe that in practice, we do not always know the exact order of the polynomial
  - Maybe 1 like a line? 3 like cubic?
  - The degree of the polynomial is a **hyperparameter**
- Experimentally, we try **several models** and pick the **best** among them



```
X, t = generate_data()
#visualize(X, t)

xs, ys = [], []
for degree in range(1, 7):
    err = learn_polynomial(X, t, degree)
    print(f'Degree {degree} has error {err}')

    xs.append(degree)
    ys.append(err)

visualize(xs, ys, is_scatter = False, tite='Degree vs Error')
```



```
y = 5 + x + 4 * x ** 2 + x ** 3                         : degree 3
Degree 1 has error 0.003505199776382123
Degree 2 has error 0.0031823079736543664
Degree 3 has error 0.0001834989378579264                ***
Degree 4 has error 0.0001827860336817794
Degree 5 has error 0.00018260965152558484
Degree 6 has error 0.00018258931984090155

y = 5 + x + 4 * x ** 2 + 5 * x ** 3 - 8 * x ** 4        : degree 4
Degree 1 has error 0.035259515944108484
Degree 2 has error 0.002934140176136343
Degree 3 has error 0.0029258075364497304
Degree 4 has error 0.00023028330775762425               ***
Degree 5 has error 0.00023010215115659195
Degree 6 has error 0.0002238670873776721
```



Error of: $y = 5 + x + 4x^2 + 5x^3 - 8x^4$

- Equation was 4th degree
- Degree 4 is the **first best**

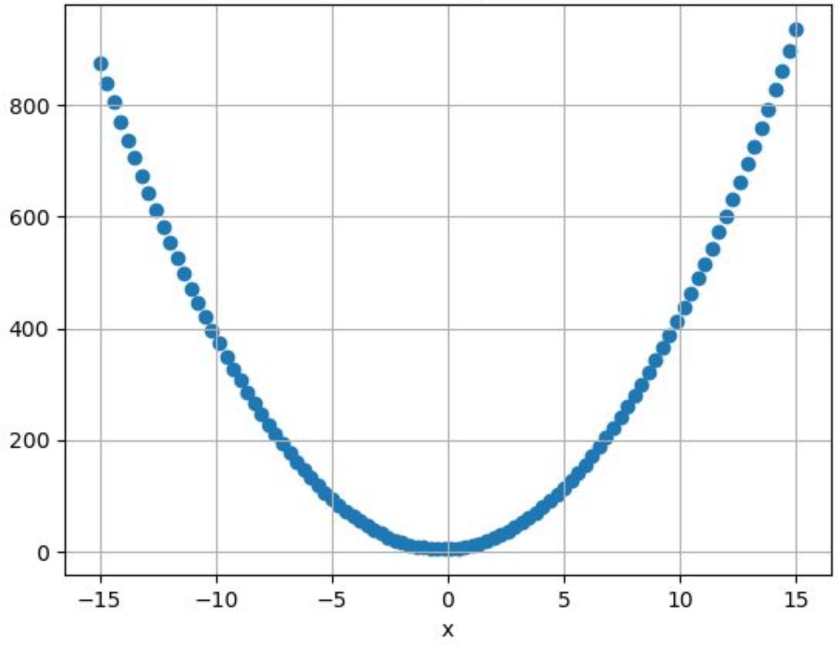
Question!

- Let's assume our dataset has 3 numeric features: A, B, C
- We want our model to use polynomial features of different degrees:
    - Degree 2 for feature A (quadratic)
    - Degree 1 for feature B (line)
    - Degree 3 for feature C (cubic)
- What is the size of the new feature vector per example?
- Ignoring scaling, and using an input of (4,6,2), what is the transformed vector?

- We only use one y-intercept for all the features. It doesn't make sense to have more
    - The total size of the new feature vector per example would be: 1 (intercept) + 2 + 1 + 3 = 7
- 1, 4^1, 4^2, 6^1, 2^1, 2^2, 2^3
    - 1 for intercept

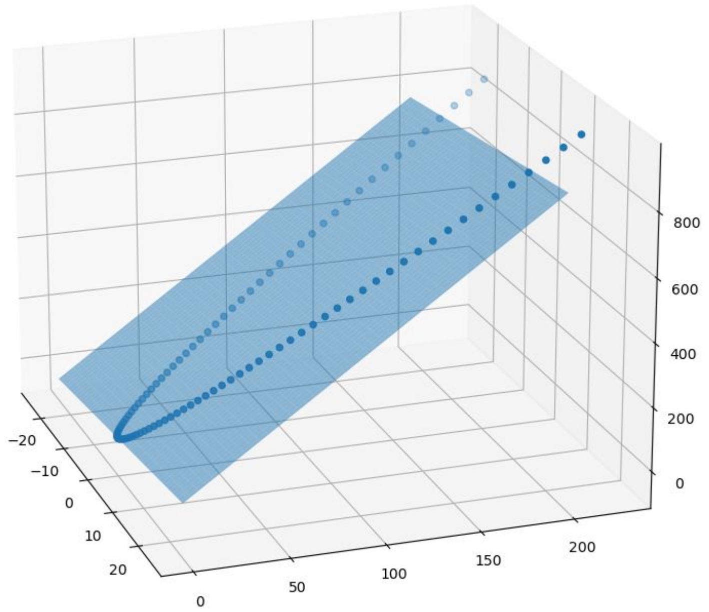
Linear Model for Non-Linear Data

- You may still not understand how non-linear data can be modeled using a linear equation
- Consider this equation:
  - $z = 2 * x + 4 * x^2 + 5$
  - Its degree is 2
- The general form for the equation of a plane is:
  - $ax + by + cz + d = 0$
  - In this case, z is the output variable
  - Rearrange
  - $z = a'x + b'y + d'$
  - So how does this help us model non-linear data using a linear equation?



Linear Model for Non-Linear Data

- The equation of a plane is itself a linear equation: $z = a'x + b'y + d'$
- The polynomial features generate non-linear data that is just **set of points on the 3D plane**
- So in terms of polynomial regression:
  - We can find the parameters W that best fit the non-linear data
  - $W = \{a', b', d'\}$
  - x and y are just real-valued input features
    - $y = x^2$



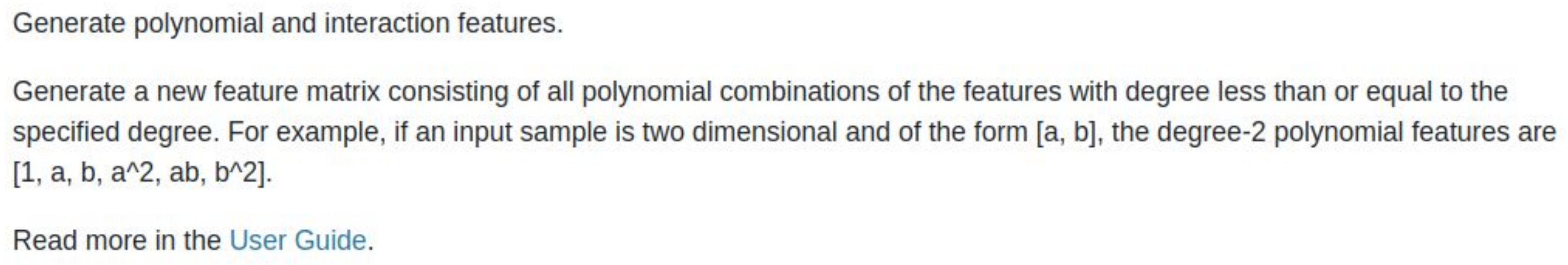
Comparison

Univariate Linear Regression	$f(x, W) = w_0 + w_1x$
Multivariate Linear Regression	$f(x, W) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$
Polynomial Linear Regression	$f(x, W) = w_0 + w_1x + w_2x^2 + w_3x^3$

Question!

- Let's assume that the data X is an Nx2 matrix (where each example has 2 features)
- If we used sklearn to transform this data using polynomial features(2), guess what is the possible output?

- While guessing is a good skill, it is always better to read the documentation
- For the question, sklearn generates ALL the combinations
  - Note in the example below: a*b has degree of 2 (the sum of the powers of a and b)



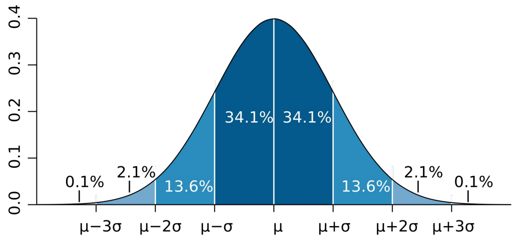
Generate polynomial and interaction features.

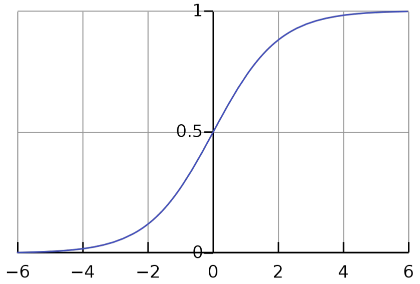
Generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2].

Read more in the User Guide.

Basis Function

- Let's generalize what we learned about polynomial regression
- Given an input vector [x1, x2, x3, ….xn], we can **transform/map** it to a set of other variables the model can work better with
- E.g. for input [x1, x2] ⇒ [f1(x1), f2(x1), f3(x1), f1(x2), f2(x2), f3(x3)]
  - The functions f1, f2, f3 are called **basis functions**
  - For example f1(x) =**polynomial**(x, 2) = x^2, f2(x) = **gaussian**(x, 1, 0), f3(x) = **sigmoid**(x)
- For a linear regression equation, this will be a **linear function** where the features are transformed in a **non-linear** way





$$S(x) = \frac{1}{1 + e^{-x}}$$

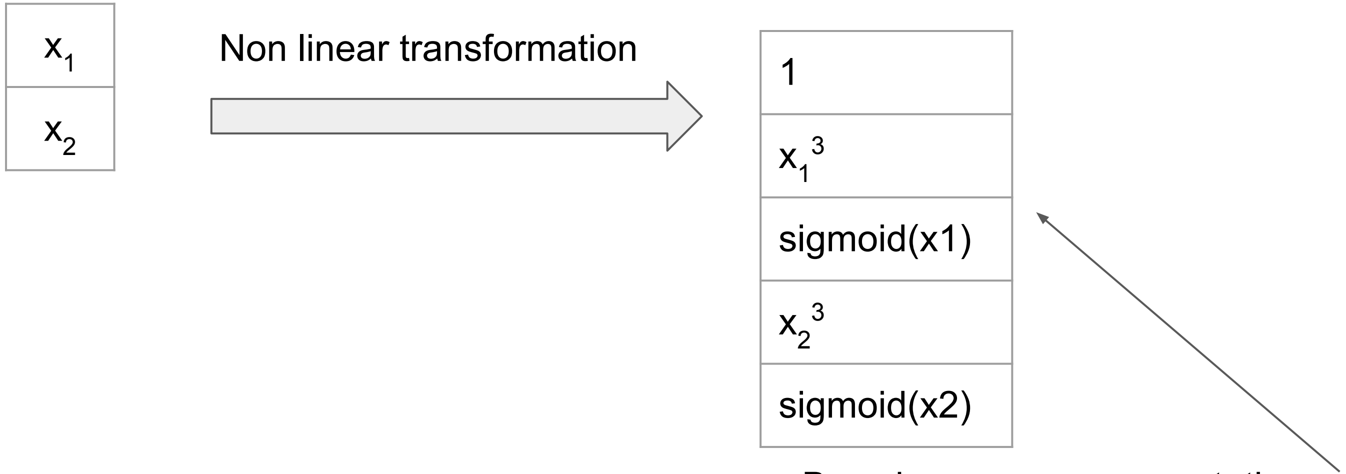
Basis Function

$$\begin{array}{|c|}\hline x_1 \\ \hline x_2 \\ \hline \end{array}$$

The raw input feature vector may be in a **2D space**, which is not well suited for machine learning

- For example, in regression, the data may not be generated from a linear relationship (i.e. a line)
- Similarly, in classification, the data may not be linearly separable (later)

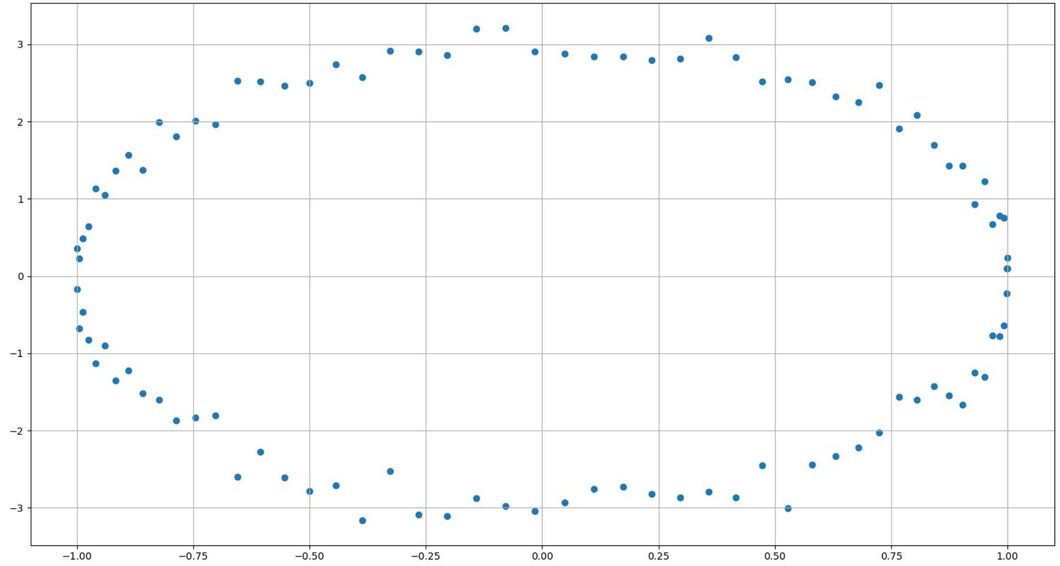
Basis Function



- By using a new representation, we can map the input feature vector to a higher dimensional space (**5D space**), where it is better suited for our machine learning model
- This kind of **mapping** is at the heart of the success of machine learning
- Especially in deep learning

Inspect this visualized data

- Imagine that we have some data, and we see the following after we visualize it:
- What do you think is the **source** function of this data?

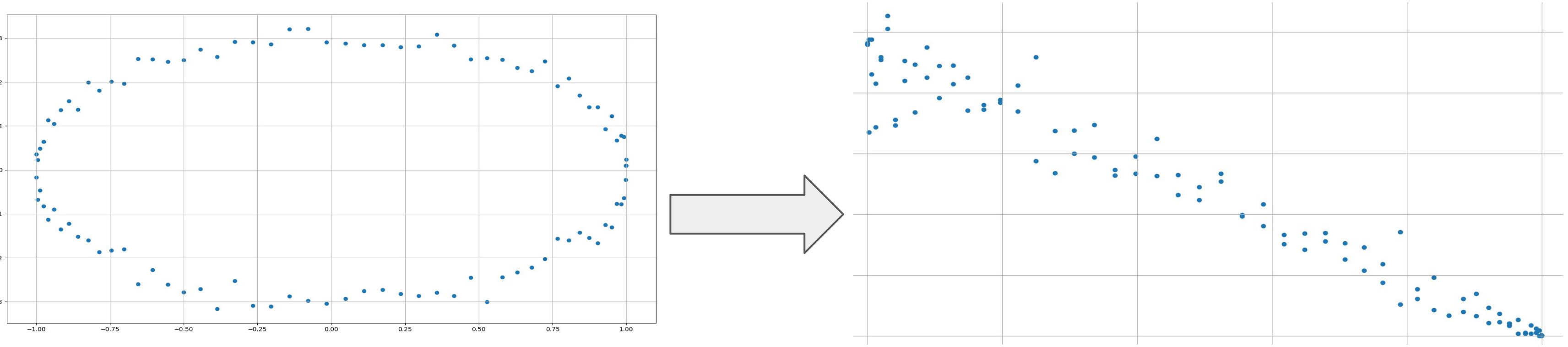


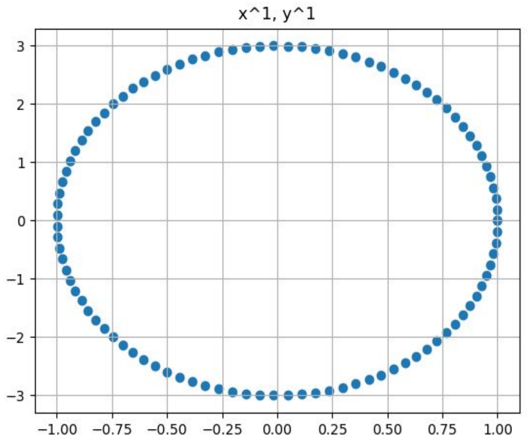
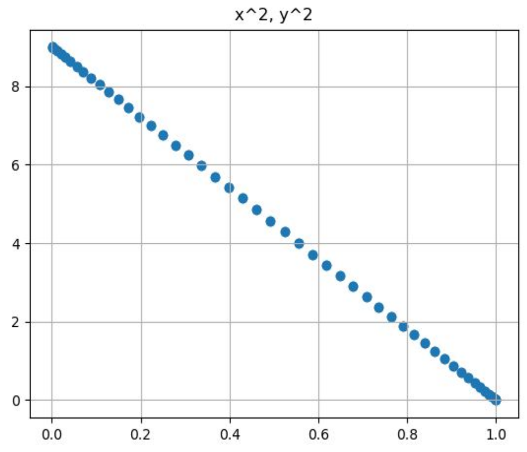
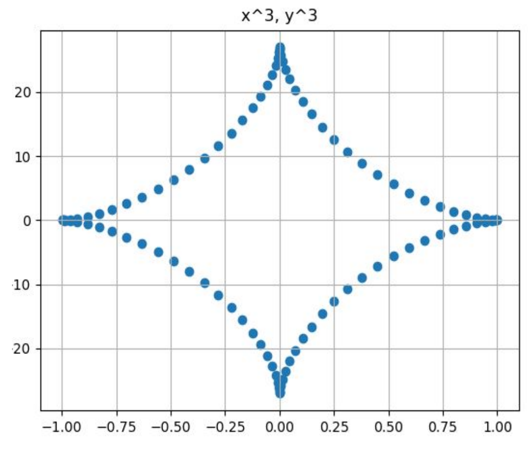
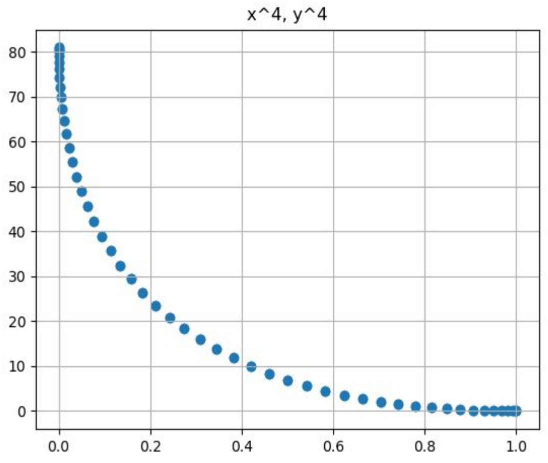
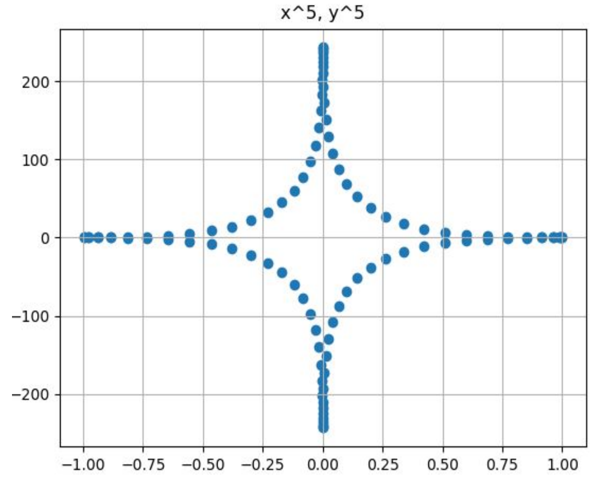
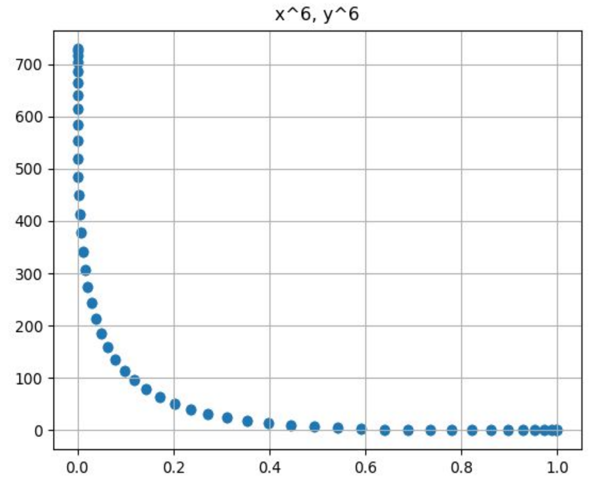
- An ellipse with:
    - Center: (0, 0)
    - a = radius x = 1
    - b = radius y = 3

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Polynomial transformation: visual effect

- With a simple transformation of each (x, y) input example to (x^2, y^2), we transform the ellipse to a line
    - Observe: positive and negative ys are now positive with y^2 (half ellipse is **mirrored**)
    - This can be a nice example in the classification later
- As you see, space transformation is a powerful trick to get different results if the model can't work well with the raw data!




x^1, y^1
3
2
1
0
−1
−2
−3
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

x^2, y^2
8
6
4
2
0
0.0 0.2 0.4 0.6 0.8 1.0

x^3, y^3
20
10
0
10
20
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

x^4, y^4
80
70
60
50
40
30
20
10
0
0.0 0.2 0.4 0.6 0.8 1.0

x^5, y^5
200
100
0
−100
−200
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

x^6, y^6
700
600
500
400
300
200
100
0
0.0 0.2 0.4 0.6 0.8 1.0

Summary

- Polynomial regression is a **form of linear** regression
  - We use it when there is a **non-linear relationship** between dependent and independent variables (modeled as an **nth degree polynomial**)
- Although polynomial regression fits a **nonlinear model** that models nonlinear data, its **equation is linear** in the unknown estimated parameters W
  - Later, when we discuss neural networks, we will see examples of non-linear regression where the equation is also non-linear
- Polynomial regression can fit with broad range of **curvature functions** and our gateway to several machine learning concepts
  - In this lecture, we found we need to try **several models** to select one
- Transforming data from one **space** to another can play a vital role in the performance of a machine learning algorithm

Relevant Resources

- Why Polynomial Regression and not Linear Regression? Article
- All you need to know about Polynomial Regression. Article
- Polynomial Regression in Python using scikit-learn. Article
- Polynomial Regression - Prof Andrew Ng
- Polynomial regression - Video
- Polynomial Regression in Python - Video
- Basis Functions: video

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."