**🧠 What is Polynomial Regression?**

**Polynomial Regression** is a type of **Supervised Learning regression** where the relationship between **input (X)** and **output (Y)** is modeled as a **polynomial** rather than a straight line.

“When data is curved, not linear, we use Polynomial Regression.”

**⚙️ Formula**

For a polynomial of degree **n**:

Y=b0+b1X+b2X2+b3X3+...+bnXnY = b\_0 + b\_1X + b\_2X^2 + b\_3X^3 + ... + b\_nX^nY=b0​+b1​X+b2​X2+b3​X3+...+bn​Xn

Where:

* **Y** = Predicted output
* **X** = Input variable
* **b₀, b₁, …, bn** = Coefficients learned by the model
* **n** = Degree of the polynomial (controls curve flexibility)

**💡 Example**

Suppose we want to predict **sales based on advertisement spend**, but the relationship is **non-linear**:

| **Ad Spend (₹k)** | **Sales (₹k)** |
| --- | --- |
| 1 | 2 |
| 2 | 5 |
| 3 | 9 |
| 4 | 16 |
| 5 | 25 |

Here, **sales increase faster than spend** — a **curve**, not a straight line.  
Linear regression would fail to capture this properly.

**📊 Visualization**

Sales (Y)

|

25| \*

20| \*

15| \*

10| \*

5| \*

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ad Spend (X)

The curve shows how Sales grows **non-linearly** with Ad Spend.

**⚙️ Python Example**

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import numpy as np

# Example data

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)

y = np.array([2, 5, 9, 16, 25])

# Transform input to polynomial features (degree 2)

poly = PolynomialFeatures(degree=2)

X\_poly = poly.fit\_transform(X)

# Train model

model = LinearRegression()

model.fit(X\_poly, y)

# Predict

y\_pred = model.predict(X\_poly)

print("Predictions:", y\_pred)

print("Mean Squared Error:", mean\_squared\_error(y, y\_pred))

**📏 Key Points**

* **Degree of polynomial** controls the curve flexibility:
  + Low degree → underfitting (too simple)
  + High degree → overfitting (too complex)
* Always visualize your data to choose the **right degree**.
* Works well when data has **curves, bends, or non-linear trends**.

**🌍 Real-World Applications**

| **Domain** | **Use Case** |
| --- | --- |
| Marketing | Predict sales vs ad spend (non-linear growth) |
| Economics | Predict GDP growth vs investments |
| Engineering | Predict stress vs load in materials |
| Healthcare | Predict drug concentration over time |
| Environment | Predict temperature or pollution trends over time |

**✅ Advantages**

* Can model **non-linear relationships**
* More flexible than Linear Regression
* Still interpretable if degree is not too high

**⚠️ Limitations**

* Sensitive to **outliers**
* High-degree polynomials → **overfitting**
* Harder to interpret coefficients for higher degrees
* Only works well with **numerical features**

**🧩 Summary Table**

| **Feature** | **Polynomial Regression** |
| --- | --- |
| **Goal** | Predict continuous value with a curve |
| **Relationship** | Non-linear (curved) |
| **Formula** | Y=b0+b1X+b2X2+...+bnXnY = b\_0 + b\_1X + b\_2X^2 + ... + b\_nX^nY=b0​+b1​X+b2​X2+...+bn​Xn |
| **Key Parameter** | Degree of the polynomial |
| **Applications** | Sales forecasting, drug concentration, material stress |
| **Advantages** | Flexible, models curves |
| **Limitations** | Overfitting, sensitive to outliers |