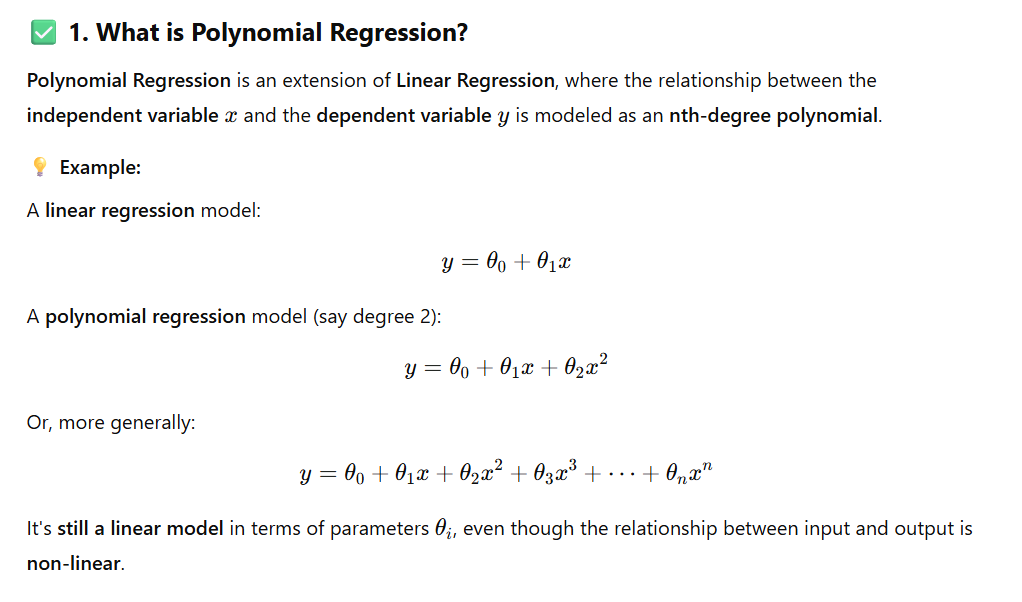
**🔍 Polynomial Linear Regression:**

[**https://www.geeksforgeeks.org/python-implementation-of-polynomial-regression/**](https://www.geeksforgeeks.org/python-implementation-of-polynomial-regression/)

**Polynomial Regression**is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modelled as an *nth-degree* polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y | x).



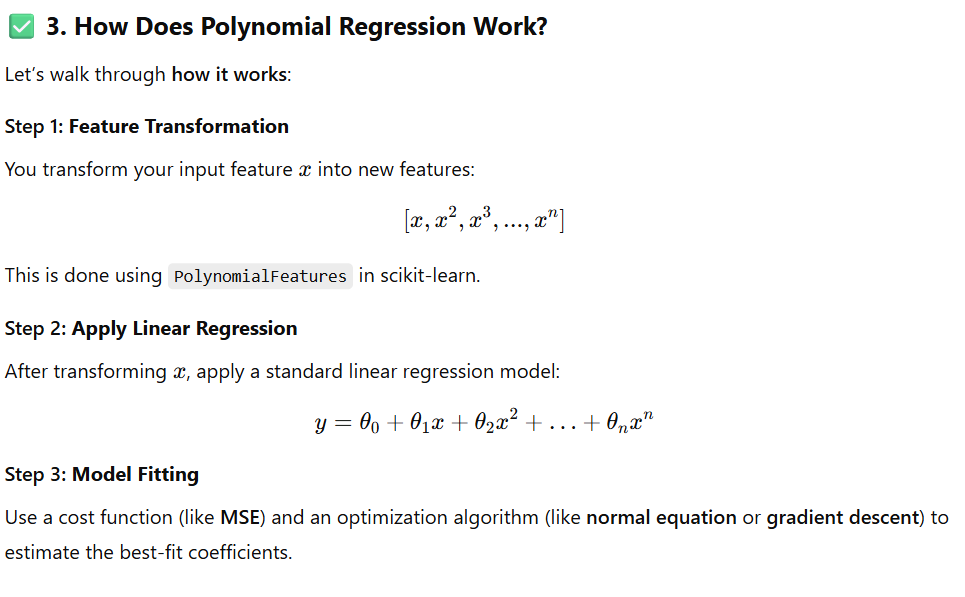
**2. When to Use Polynomial Regression?**

Use **polynomial regression** when:

* The data shows a **curved** or **non-linear** relationship between input and output.
* A straight line **doesn’t fit** the data well (i.e., high residuals in linear regression).
* You suspect the dependent variable depends on higher powers of the independent variable.

**📈 Common Use Cases:**

* Growth curves (e.g. population, bacteria).
* Economics trends.
* Physics/chemistry (e.g. motion under acceleration).
* Any data with a **parabolic or wave-like shape**.



**Example: Polynomial Regression vs Linear Regression**

We'll use **synthetic non-linear data** that mimics real-world curvature (e.g., y=x2+noisey = x^2 + noisey=x2+noise) to show the difference clearly.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.model\_selection import train\_test\_split

np.random.seed(42)

X = np.linspace(0, 10, 100).reshape(-1, 1)

y = 3 \* X.squeeze()\*\*2 + 2 \* X.squeeze() + 1 + np.random.randn(100) \* 10 # y = 3x² + 2x + 1 + noise

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

lin\_model = LinearRegression()

lin\_model.fit(X\_train, y\_train)

# Predict

y\_pred\_lin = lin\_model.predict(X\_test)

# Evaluation

mse\_lin = mean\_squared\_error(y\_test, y\_pred\_lin)

r2\_lin = r2\_score(y\_test, y\_pred\_lin)

poly = PolynomialFeatures(degree=2)

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

X\_test\_poly = poly.transform(X\_test)

poly\_model = LinearRegression()

poly\_model.fit(X\_train\_poly, y\_train)

# Predict

y\_pred\_poly = poly\_model.predict(X\_test\_poly)

# Evaluation

mse\_poly = mean\_squared\_error(y\_test, y\_pred\_poly)

r2\_poly = r2\_score(y\_test, y\_pred\_poly)

plt.figure(figsize=(12, 6))

# Scatter real data

plt.scatter(X\_test, y\_test, color='blue', label='Actual Test Data')

# Plot linear regression line

plt.plot(X\_test, y\_pred\_lin, color='red', label='Linear Regression')

# Plot polynomial regression curve

X\_curve = np.linspace(0, 10, 100).reshape(-1, 1)

X\_curve\_poly = poly.transform(X\_curve)

y\_curve\_poly = poly\_model.predict(X\_curve\_poly)

plt.plot(X\_curve, y\_curve\_poly, color='green', label='Polynomial Regression (degree=2)')

plt.title("Linear vs Polynomial Regression")

plt.xlabel("X")

plt.ylabel("y")

plt.legend()

plt.grid(True)

plt.show()

print("🔍 Evaluation Results:")

print(f"Linear Regression MSE: {mse\_lin:.2f}, R² Score: {r2\_lin:.4f}")

print(f"Polynomial Regression MSE: {mse\_poly:.2f}, R² Score: {r2\_poly:.4f}")

**Interpretation**

* **Linear Regression** may show **underfitting**: it can't capture the curvature.
* **Polynomial Regression** fits the data better with lower MSE and higher R2R^2R2.

**Overfitting Vs Under-fitting**

While dealing with the polynomial regression one thing that we face is the problem of [overfitting](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/) this happens because while we increase the order of the polynomial regression to achieve better and better performance model gets overfit on the data and does not perform on the new data points.

Due to this reason only while using the polynomial regression, do we try to penalize the weights of the model to regularize the effect of the overfitting problem. [Regularization](https://www.geeksforgeeks.org/regularization-in-machine-learning/) techniques like [Lasso regression](https://www.geeksforgeeks.org/implementation-of-lasso-regression-from-scratch-using-python/) and [Ridge regression](https://www.geeksforgeeks.org/implementation-of-ridge-regression-from-scratch-using-python/) methodologies are used whenever we deal with a situation in which the model may overfit the data at hand.

**Bias Vs Variance Tradeoff**

This technique is the generalization of the approach that is used to avoid the problem of overfitting and underfitting. Here as well this technique helps us to avoid the problem of overfitting by helping us select the appropriate value for the degree of the polynomial we are trying to fit our data on. For example, this is achieved when after increasing the degree of polynomial after a certain level the gap between the training and the validation metrics starts increasing.

**Application of Polynomial Regression**

The reason behind the vast use cases of the polynomial regression is that approximately all of the real-world data is non-linear in nature and hence when we fit a non-linear model on the data or a curvilinear regression line then the results that we obtain are far better than what we can achieve with the standard linear regression. Some of the use cases of the Polynomial regression are as stated below:

* The growth rate of tissues.
* Progression of disease epidemics
* Distribution of carbon isotopes in lake sediments

**Advantages & Disadvantages of using Polynomial Regression**

**Advantages of using Polynomial Regression**

* A broad range of functions can be fit under it.
* Polynomial basically fits a wide range of curvatures.
* Polynomial provides the best approximation of the relationship between dependent and independent variables.

**Disadvantages of using Polynomial Regression**

* These are too sensitive to outliers.
* The presence of one or two [outliers](https://www.geeksforgeeks.org/machine-learning-outlier/) in the data can seriously affect the results of nonlinear analysis.
* In addition, there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.