NumPy: For efficient generation and manipulation of numerical data, such as creating 100 data points for the x-axis.

Matplotlib: For plotting charts to visualize and compare the fitting results of different models.

Scikit-learn:

1. PolynomialFeatures: Transforms the original data X into a higher-degree polynomial feature set.
2. LinearRegression: Uses a linear regression model to train on these polynomial features, finding the best-fit curve.

Model Comparison and Analysis

The code sets up and compares four different models with polynomial degrees of 2, 5, 50, and 10^5.

Degree 2 (Green Line) & Degree 5 (Purple Line):

These models are underfitting. Because their polynomial degree is too low, they cannot fully capture the complexity of the original function's curve. The fitted lines appear smooth but fail to accurately pass through all the data points. Lower-degree polynomial models generally have better generalization ability, meaning their predictions are more stable for unseen data.

Degree 30 (Black Dashed Line):

When the degree is increased to 30, the model begins to fit the data closely. It strikes a good balance, capturing the overall trend of the data without being overly complex.

Degree 10^5 (Red Line):

This extremely high-degree model is a classic case of overfitting. It attempts to perfectly pass through every single data point, resulting in severe oscillations between the points. This model has excellent performance on the training data but would fail dramatically on any new, unseen data.

Conclusion

This code clearly demonstrates that in function approximation, a higher model complexity is not always better. An appropriate model should strike a balance between underfitting and overfitting, effectively capturing the underlying patterns in the data while maintaining stable and reliable predictive capabilities.