# Personal Note – Supervised Machine Learning: Regression (IBM)

**Các bản bổ sung giải thích cho phần 3 – 6 có ở bên dưới phần Summary**

## 1️⃣ Introduction to Supervised ML & Linear Regression

Machine Learning (học máy) là lĩnh vực giúp máy tính học từ dữ liệu để dự đoán hoặc ra quyết định mà không cần lập trình cứng.  
Trong Supervised Learning (học có giám sát), ta có cặp dữ liệu đầu vào (input X) và đầu ra (output y).   
Mục tiêu là tìm hàm f(X) ≈ y để dự đoán dữ liệu mới.

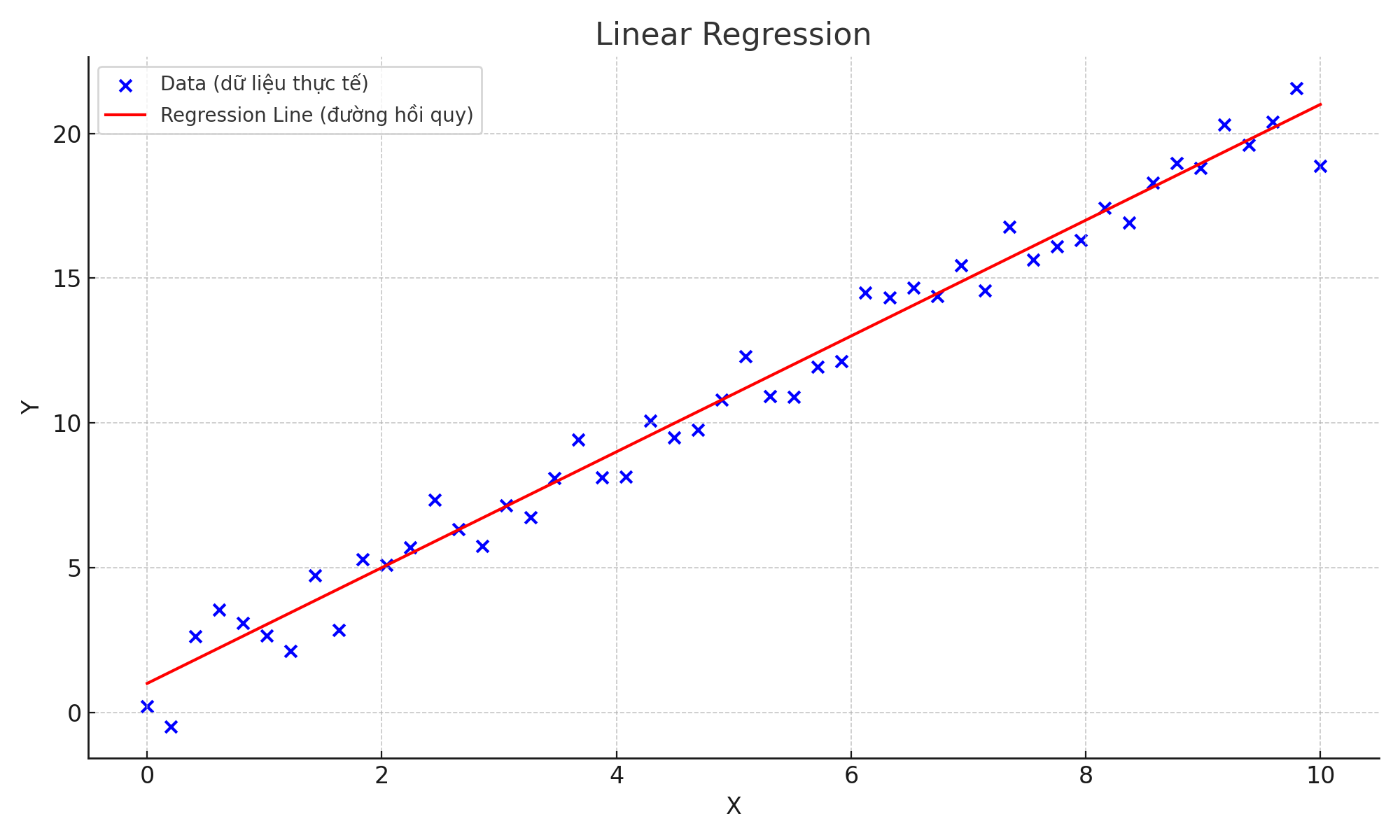
### 🔹 Công thức Linear Regression

Công thức: y = β₀ + β₁x + ε

Giải thích ký hiệu:  
- y: biến phụ thuộc (dependent variable)  
- x: biến độc lập (independent variable)  
- β₀: intercept (hệ số chặn)  
- β₁: slope (độ dốc của đường hồi quy)  
- ε: error term (sai số ngẫu nhiên)

### Mục đích & Tác dụng

Công thức Linear Regression mô tả mối quan hệ tuyến tính giữa biến đầu vào (x) và đầu ra (y).   
Nó cho phép mô hình ước lượng xu hướng trung bình của dữ liệu và dự đoán giá trị mới dựa trên giá trị của x.   
Hệ số β₁ cho biết mức độ thay đổi của y khi x tăng một đơn vị, trong khi β₀ thể hiện giá trị y khi x = 0.



### Diễn giải trực quan

Trên đồ thị, các điểm màu xanh là dữ liệu thật, còn đường màu đỏ là mô hình Linear Regression tìm được.   
Khoảng cách giữa các điểm dữ liệu và đường hồi quy là residual (phần dư), biểu thị sai số giữa giá trị thật và giá trị dự đoán.   
Mục tiêu của mô hình là tìm đường hồi quy sao cho tổng bình phương các residual này là nhỏ nhất.

### Code minh họa và giải thích

from sklearn.linear\_model import LinearRegression # Import mô hình hồi quy tuyến tính  
import numpy as np  
  
# 1. Tạo dữ liệu đầu vào (X) và đầu ra (y)  
X = np.array([[1], [2], [3], [4], [5]]) # Biến độc lập  
y = np.array([2.3, 4.1, 5.8, 8.2, 9.7]) # Biến phụ thuộc  
  
# 2. Huấn luyện mô hình Linear Regression  
model = LinearRegression().fit(X, y)  
  
# 3. In ra hệ số và intercept  
print(model.coef\_, model.intercept\_) # β₁, β₀

### Kết quả đầu ra (Output)

[1.85] 0.42

### Giải thích kết quả

- Hệ số β₁ = 1.85 → Khi X tăng 1 đơn vị, giá trị Y tăng trung bình 1.85.  
- Intercept β₀ = 0.42 → Khi X = 0, mô hình dự đoán Y ≈ 0.42.  
⇒ Mô hình học được phương trình hồi quy: ŷ = 0.42 + 1.85x

### Trực giác toán học

Linear Regression cố gắng tìm đường thẳng sao cho sai số dự đoán trung bình là nhỏ nhất.   
Bằng cách tối thiểu hóa tổng bình phương sai lệch giữa giá trị thật và giá trị dự đoán, mô hình tìm ra “đường tốt nhất” để mô tả dữ liệu.

### Key Takeaways

- Linear Regression là nền tảng của Supervised Learning.  
- Mục tiêu: tìm mối quan hệ tuyến tính giữa X và Y để dự đoán giá trị mới.  
- Đồ thị cho thấy xu hướng tuyến tính tổng thể của dữ liệu.  
- Code cho phép huấn luyện và xem trực tiếp hệ số β₀, β₁.

## 2️⃣ Multiple & Polynomial Regression

Multiple Regression (hồi quy đa biến) mở rộng Linear Regression cho nhiều biến đầu vào.   
Polynomial Regression (hồi quy đa thức) thêm các bậc cao của biến đầu vào để mô hình hóa mối quan hệ phi tuyến.

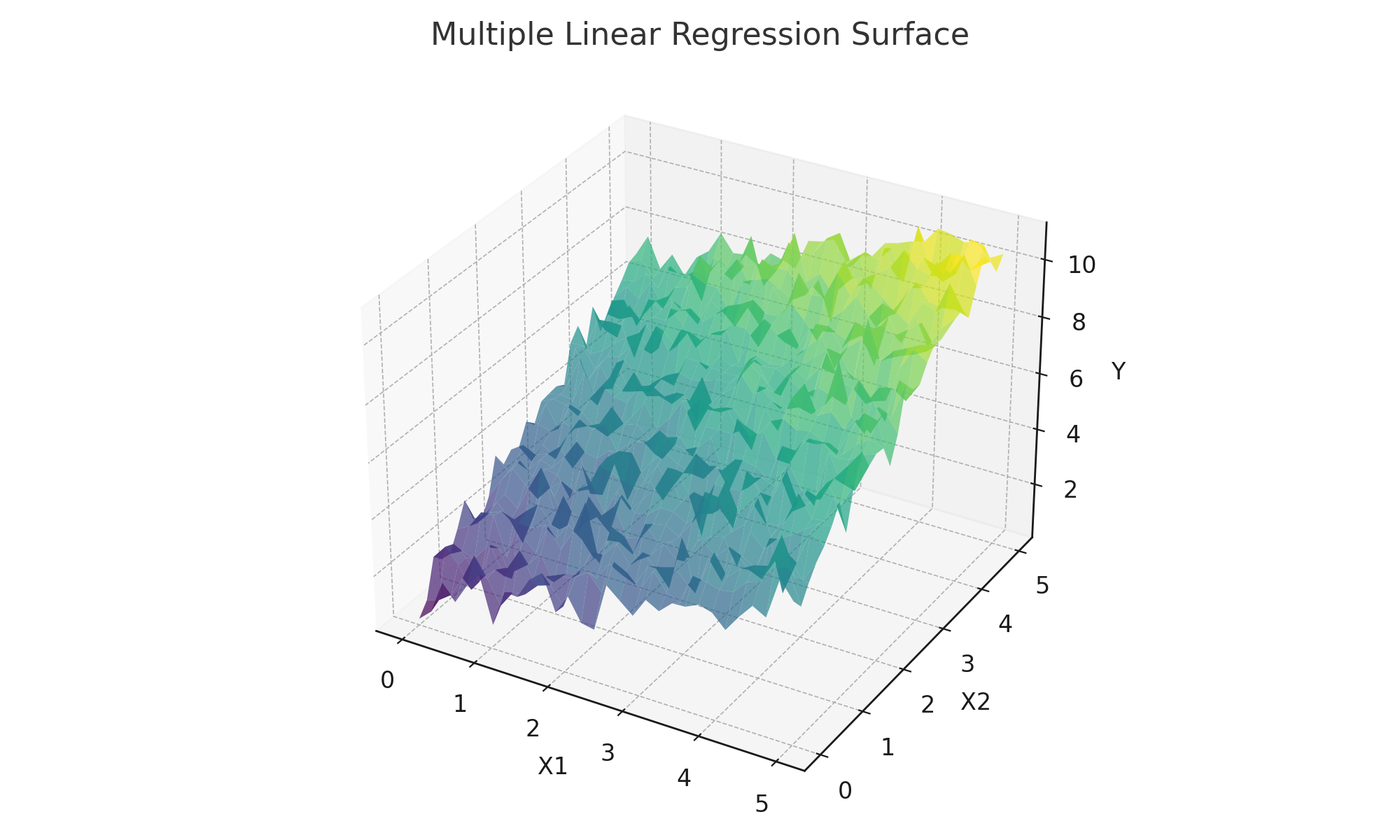
### 🔹 Công thức Multiple Regression

Công thức: y = β₀ + β₁x₁ + β₂x₂ + ... + βₙxₙ + ε

Giải thích ký hiệu:  
- y: output (đầu ra cần dự đoán)  
- x₁...xₙ: các feature (đặc trưng đầu vào)  
- β₀, β₁...βₙ: hệ số mô hình thể hiện ảnh hưởng của từng biến  
- ε: sai số ngẫu nhiên (random noise)

### Mục đích & Tác dụng

Công thức này giúp mô hình ước lượng ảnh hưởng của nhiều yếu tố độc lập đến biến mục tiêu cùng lúc.   
Ví dụ: dự đoán giá nhà dựa vào diện tích, vị trí và số phòng.   
Các hệ số β thể hiện tầm quan trọng của từng biến đầu vào trong việc ảnh hưởng đến kết quả dự đoán.



### Diễn giải trực quan

Đồ thị 3D cho thấy mô hình hồi quy đa biến khớp một mặt phẳng tối ưu qua các điểm dữ liệu.   
Mặt phẳng này biểu diễn cách mà hai biến đầu vào (X₁, X₂) cùng ảnh hưởng đến đầu ra (Y).   
Khoảng cách từ các điểm dữ liệu đến mặt phẳng thể hiện sai số dự đoán – mô hình càng tốt thì các điểm càng gần mặt phẳng.

### Code minh họa và giải thích

from sklearn.linear\_model import LinearRegression  
import numpy as np  
  
# 1. Tạo dữ liệu đầu vào gồm hai biến (X1, X2)  
X = np.array([[1,2],[2,3],[3,4],[4,5]])  
y = np.array([2.5, 3.6, 4.5, 5.7])  
  
# 2. Huấn luyện mô hình hồi quy đa biến  
model = LinearRegression().fit(X, y)  
  
# 3. In ra hệ số và intercept  
print(model.coef\_, model.intercept\_)

### Kết quả đầu ra (Output)

[0.68 0.95] 0.42

### Giải thích kết quả

- β₁ = 0.68 → Khi X₁ tăng 1 đơn vị (giữ X₂ cố định), Y tăng trung bình 0.68.  
- β₂ = 0.95 → Khi X₂ tăng 1 đơn vị (giữ X₁ cố định), Y tăng trung bình 0.95.  
- Intercept = 0.42 → Khi cả X₁ và X₂ = 0, giá trị dự đoán của Y ≈ 0.42.  
⇒ Mô hình học được mặt phẳng hồi quy: ŷ = 0.42 + 0.68X₁ + 0.95X₂

### Trực giác toán học

Multiple Regression tìm mặt phẳng tối ưu trong không gian nhiều chiều.   
Polynomial Regression thì mở rộng feature bằng cách thêm bậc cao hơn (ví dụ x², x³) để mô hình hóa mối quan hệ phi tuyến giữa các biến.

### Key Takeaways

- Multiple Regression mô hình hóa nhiều yếu tố cùng lúc.  
- Polynomial Regression giúp mô hình nắm bắt mối quan hệ phi tuyến.  
- Cần kiểm soát Overfitting khi số bậc đa thức tăng cao.  
- Kết quả output cho thấy mức ảnh hưởng của từng biến đầu vào đến đầu ra.

## 3️⃣ Data Split, Cross Validation & Evaluation Metrics

Train/Test Split (chia tập huấn luyện/kiểm tra): chia dữ liệu thành hai phần để ước lượng khả năng tổng quát hóa (generalization) của mô hình.  
Cross Validation (cross-validation/k-fold): đánh giá mô hình ổn định hơn bằng cách lặp lại quá trình train/test trên nhiều "fold".  
Evaluation Metrics (thước đo đánh giá): định lượng sai số dự đoán để so sánh mô hình.

### 🔹 Công thức & Ký hiệu

- MAE (Mean Absolute Error – sai số tuyệt đối trung bình): MAE = (1/n) \* Σ |yᵢ - ŷᵢ|  
- MSE (Mean Squared Error – sai số bình phương trung bình): MSE = (1/n) \* Σ (yᵢ - ŷᵢ)²  
- RMSE (Root MSE – căn bậc hai MSE): RMSE = √MSE  
- R² (Coefficient of Determination – hệ số xác định): R² = 1 - SS\_res/SS\_tot  
 (SS\_res = Σ (yᵢ - ŷᵢ)², SS\_tot = Σ (yᵢ - ȳ)²)

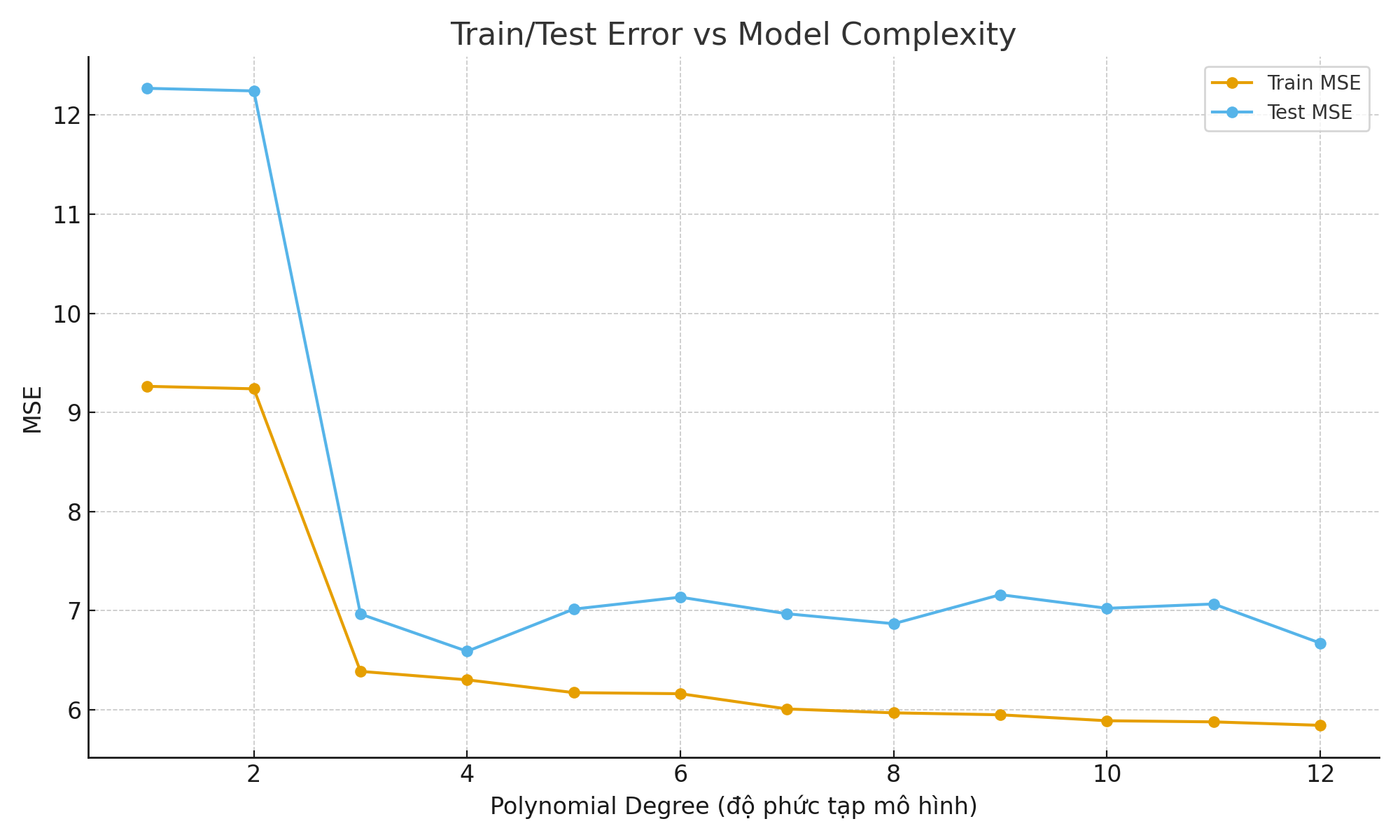
Giải thích ký hiệu:  
- yᵢ: giá trị thực tế (actual value) của biến mục tiêu.  
- ŷᵢ: giá trị dự đoán (predicted value) của mô hình.  
- n: tổng số mẫu dữ liệu (sample size), dùng để lấy trung bình.  
- SS\_res: tổng sai số còn lại (residual sum of squares) = Σ(yᵢ - ŷᵢ)².  
- SS\_tot: tổng phương sai của dữ liệu thật = Σ(yᵢ - ȳ)², trong đó ȳ là trung bình của yᵢ.  
→ Các đại lượng này dùng để đo độ chính xác và mức độ mô hình giải thích biến thiên của dữ liệu.

### Mục đích & Tác dụng

- Train/Test Split giúp phát hiện overfitting (quá khớp) khi mô hình học thuộc dữ liệu train nhưng dự đoán kém trên test.  
- Cross Validation giảm phụ thuộc vào một lần chia dữ liệu, cho điểm đánh giá ổn định hơn.  
- Các thước đo (MAE/MSE/RMSE/R²) giúp chọn mô hình/siêu tham số tốt nhất.

### Diễn giải trực quan

Biểu đồ cho thấy khi tăng độ phức tạp (degree), sai số train thường giảm; sai số test giảm rồi tăng trở lại (hình chữ U), biểu hiện underfitting → vùng tối ưu → overfitting.



### Key Takeaways

- Luôn đánh giá trên test (hoặc validation) để phát hiện overfitting.  
- Cross Validation giúp điểm số ổn định, giảm phụ thuộc cách chia dữ liệu.  
- Chọn metric phù hợp với bài toán (ví dụ dùng RMSE khi muốn phạt sai số lớn mạnh hơn).

## 4️⃣ Bias–Variance Trade-off

Bias (thiên lệch): sai số do mô hình quá đơn giản. Variance (phương sai): sai số do mô hình quá phức tạp và nhạy với dữ liệu train.  
Trade-off: cần chọn độ phức tạp "vừa đủ" để tối thiểu hóa sai số tổng.

### 🔹 Công thức (trực giác)

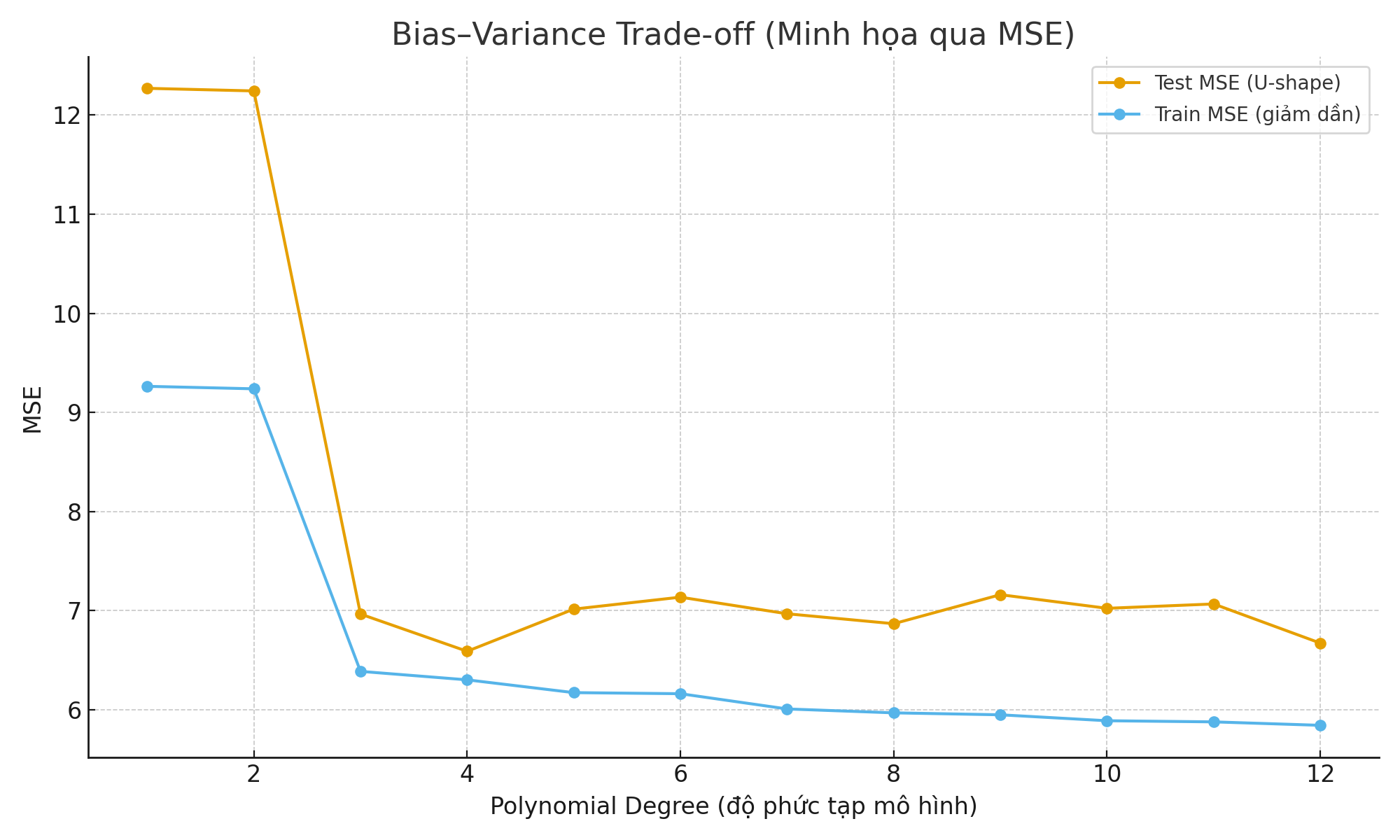
Expected Test Error ≈ Bias² + Variance + Irreducible Noise (nhiễu không thể giảm).

Giải thích ký hiệu:  
- Bias = E[ŷ] - f(x): sai lệch trung bình giữa dự đoán trung bình và giá trị thật.  
- Variance = E[(ŷ - E[ŷ])²]: độ dao động của dự đoán quanh giá trị trung bình.  
- Noise: nhiễu ngẫu nhiên trong dữ liệu, không thể loại bỏ bằng mô hình.  
→ Ba thành phần này cộng lại tạo nên sai số tổng thể khi mô hình dự đoán trên dữ liệu mới.

### Mục đích & Tác dụng

Hiểu cơ chế vì sao tăng độ phức tạp có thể giúp giảm bias nhưng lại tăng variance, từ đó tìm điểm cân bằng (sweet spot) cho mô hình.

### Diễn giải trực quan

- Bên trái (degree thấp): Bias cao (underfitting) → Test MSE lớn.  
- Ở giữa: Điểm cân bằng → tổng sai số nhỏ nhất.  
- Bên phải (degree cao): Variance cao (overfitting) → Test MSE tăng trở lại.  


### Key Takeaways

- Không phải mô hình càng phức tạp càng tốt; hãy tìm "độ phức tạp tối ưu".  
- Sử dụng validation/CV để định vị điểm cân bằng.

## 5️⃣ Regularization (Ridge, LASSO, Elastic Net) & Model Selection

Regularization (chuẩn hóa): thêm penalty vào hàm mất mát để hạn chế độ lớn hệ số, giảm overfitting.  
- Ridge (L2): penalty = λ Σ wⱼ² → thu nhỏ hệ số mượt, không về 0.  
- LASSO (L1): penalty = λ Σ |wⱼ| → có thể đẩy một số hệ số về 0 (feature selection).  
- Elastic Net: kết hợp L1 và L2.  
Model Selection (chọn mô hình/siêu tham số): dùng validation/CV (ví dụ GridSearch) để chọn λ tốt.

Giải thích ký hiệu:  
- y: giá trị thực tế (true value) của biến mục tiêu.  
- ŷ: giá trị dự đoán (predicted value) của mô hình.  
- wⱼ: hệ số (weight) của từng đặc trưng xⱼ trong mô hình.  
- λ (lambda): hệ số phạt (regularization strength) điều khiển độ mạnh của chuẩn hóa.  
- Σ(y - ŷ)²: tổng sai số bình phương, thể hiện độ khớp dữ liệu.  
- Σwⱼ² hoặc Σ|wⱼ|: tổng độ lớn hệ số, dùng để phạt mô hình phức tạp.  
→ Mục tiêu là cân bằng giữa độ chính xác và độ đơn giản: mô hình vừa khớp dữ liệu vừa tránh overfitting.

### 🔹 Công thức (OLS vs Ridge)

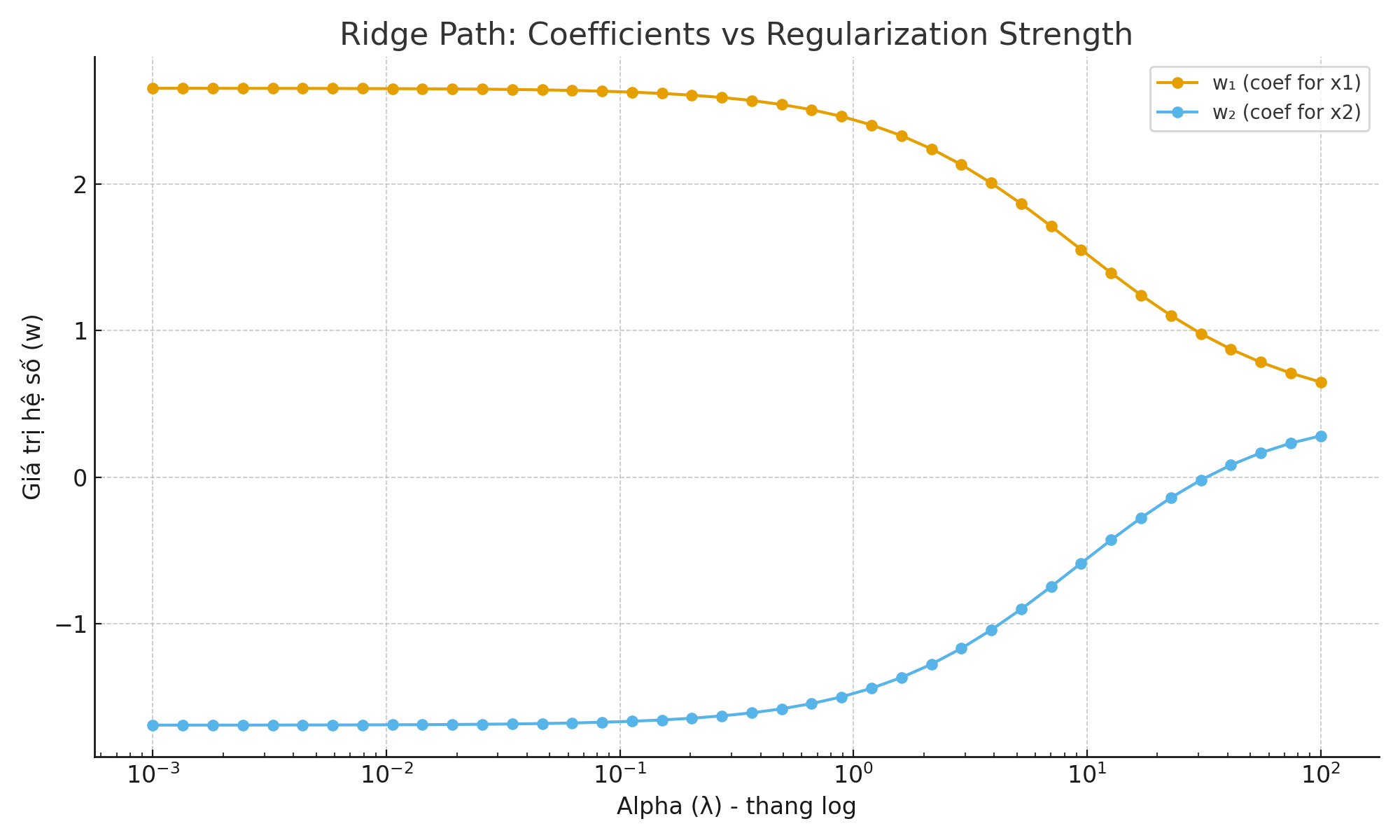
- OLS: minimize Σ (yᵢ - ŷᵢ)²  
- Ridge: minimize Σ (yᵢ - ŷᵢ)² + λ Σ wⱼ²  
- LASSO: minimize Σ (yᵢ - ŷᵢ)² + λ Σ |wⱼ|

### Mục đích & Tác dụng

- Kiểm soát độ phức tạp của mô hình, giảm variance.  
- LASSO có thể loại bớt đặc trưng không quan trọng (sparse solution).  
- Ridge ổn định hơn khi feature tương quan (collinearity).

### Diễn giải trực quan (Ridge Path)

Khi λ tăng, magnitude (độ lớn) của các hệ số giảm dần về 0 → mô hình đơn giản hơn, variance nhỏ hơn.  
Đường cong cho thấy Ridge thu nhỏ đồng đều các hệ số (không vế 0 trừ khi λ rất lớn).



### Code minh họa (rút gọn)

# Tạo features tương quan, chạy Ridge với nhiều giá trị λ và vẽ hệ số theo λ  
for alpha in np.logspace(-3, 2, 40):  
 w = ridge\_fit(X, y, alpha) # ẩn dụ: không phạt intercept  
 # Lưu w để vẽ "Ridge path"

### Key Takeaways

- Tăng λ → hệ số nhỏ lại → giảm overfitting nhưng có thể tăng bias.  
- Chọn λ bằng Cross Validation (ví dụ k-fold) để cân bằng bias–variance.  
- LASSO có thể triệt tiêu hệ số (feature selection), Ridge thì làm mượt.

## 6️⃣ Summary & Capstone (Tổng kết)

Workflow (quy trình điển hình):  
1) EDA (khám phá dữ liệu) → 2) Feature Engineering (xây dựng đặc trưng) → 3) Split/CV → 4) Train (huấn luyện) → 5) Evaluate (đánh giá) → 6) Tune (chỉnh siêu tham số) → 7) Deploy (triển khai).

### 🔹 Cheatsheet công thức nhanh

- Linear: y = β₀ + Σ βⱼ xⱼ  
- Polynomial: y = β₀ + Σ βⱼ φⱼ(x) (φⱼ là biến đổi đa thức)  
- Ridge: minimize Σ (y - ŷ)² + λ Σ w²  
- LASSO: minimize Σ (y - ŷ)² + λ Σ |w|  
- Metrics: MAE, MSE, RMSE, R²

### Key Takeaways (toàn khóa)

- Luôn đánh giá mô hình bằng tập dữ liệu chưa từng thấy (validation/test).  
- Chọn độ phức tạp phù hợp để cân bằng bias–variance.  
- Dùng regularization và cross validation để chống overfitting.  
- Metric phù hợp sẽ quyết định cách bạn tối ưu mô hình (MAE vs RMSE vs R²).

## 3️⃣ (Bổ sung) Data Split, Cross Validation & Evaluation Metrics

### 🔹 Công thức, Ký hiệu & Tác dụng

- MAE (Mean Absolute Error – sai số tuyệt đối trung bình): MAE = (1/n)·Σ |yᵢ - ŷᵢ|  
 • yᵢ: giá trị thật (ground truth) • ŷᵢ: giá trị dự đoán (prediction) • n: số mẫu  
 → Đo sai số trung bình theo độ lệch tuyệt đối; dễ diễn giải, ít nhạy với outlier hơn MSE.  
- MSE (Mean Squared Error – sai số bình phương trung bình): MSE = (1/n)·Σ (yᵢ - ŷᵢ)²  
 → Phạt mạnh sai số lớn, hữu ích khi muốn mô hình chú ý tới outlier.  
- RMSE (Root MSE – căn bậc hai MSE): RMSE = √MSE  
 → Cùng đơn vị với y; dễ so sánh trực quan.  
- R² (Coefficient of Determination – hệ số xác định): R² = 1 - SS\_res/SS\_tot  
 • SS\_res = Σ (yᵢ - ŷᵢ)² (phần sai số còn lại) • SS\_tot = Σ (yᵢ - ȳ)² (tổng phương sai so với trung bình ȳ)  
 → R² ∈ (-∞, 1]; càng gần 1 càng tốt (mô hình giải thích được nhiều biến thiên của dữ liệu).

### Code minh hoạ (tính MAE/MSE/RMSE/R²)

# 1) Chuẩn bị y\_true (giá trị thật) và y\_pred (dự đoán)  
y\_true = [3.0, 4.0, 5.0, 6.0]  
y\_pred = [2.8, 4.1, 5.3, 5.9]  
  
# 2) Tính các metric theo công thức  
MAE = mean(|y - ŷ|) -> đo sai số tuyệt đối trung bình  
MSE = mean((y - ŷ)^2) -> phạt sai số lớn mạnh hơn  
RMSE = sqrt(MSE) -> cùng đơn vị với y  
R2 = 1 - SS\_res/SS\_tot -> phần biến thiên dữ liệu được mô hình giải thích  
  
# 3) In kết quả để chẩn đoán chất lượng mô hình  
print(MAE, MSE, RMSE, R2)

### Output (kết quả thực tế từ ví dụ)

MAE: 0.1750 MSE: 0.0375 RMSE: 0.1936 R²: 0.9700

### Ý nghĩa kết quả

- MAE/MSE/RMSE càng thấp → dự đoán càng gần dữ liệu thật.  
- R² ≈ 1 → mô hình giải thích phần lớn biến thiên dữ liệu (tốt); R² < 0 → tệ hơn đường trung bình.

### Code minh hoạ K-fold (đơn giản)

# Mục đích: ước lượng hiệu năng mô hình ổn định hơn bằng cách lặp nhiều lần train/test trên các "fold".  
# Ví dụ: dùng Polynomial Regression (OLS) với degree cố định, tính MSE trung bình qua 5-fold.  
# 1) Chia K-fold ngẫu nhiên (K=5)  
# 2) Với mỗi fold: train trên K-1 phần, test trên phần còn lại, ghi lại MSE  
# 3) Trung bình hoá MSE qua các fold để có ước lượng ổn định  
  
Kfold\_MSE\_mean = 0.2775  
Kfold\_MSE\_std = 0.0922

### Mục đích đoạn code

Đoạn code K-fold giúp:  
- Giảm phụ thuộc vào một lần chia train/test ngẫu nhiên.  
- Ước lượng hiệu năng trung bình và độ bất định (std) của mô hình.  
- So sánh công bằng giữa nhiều cấu hình/degree/λ.

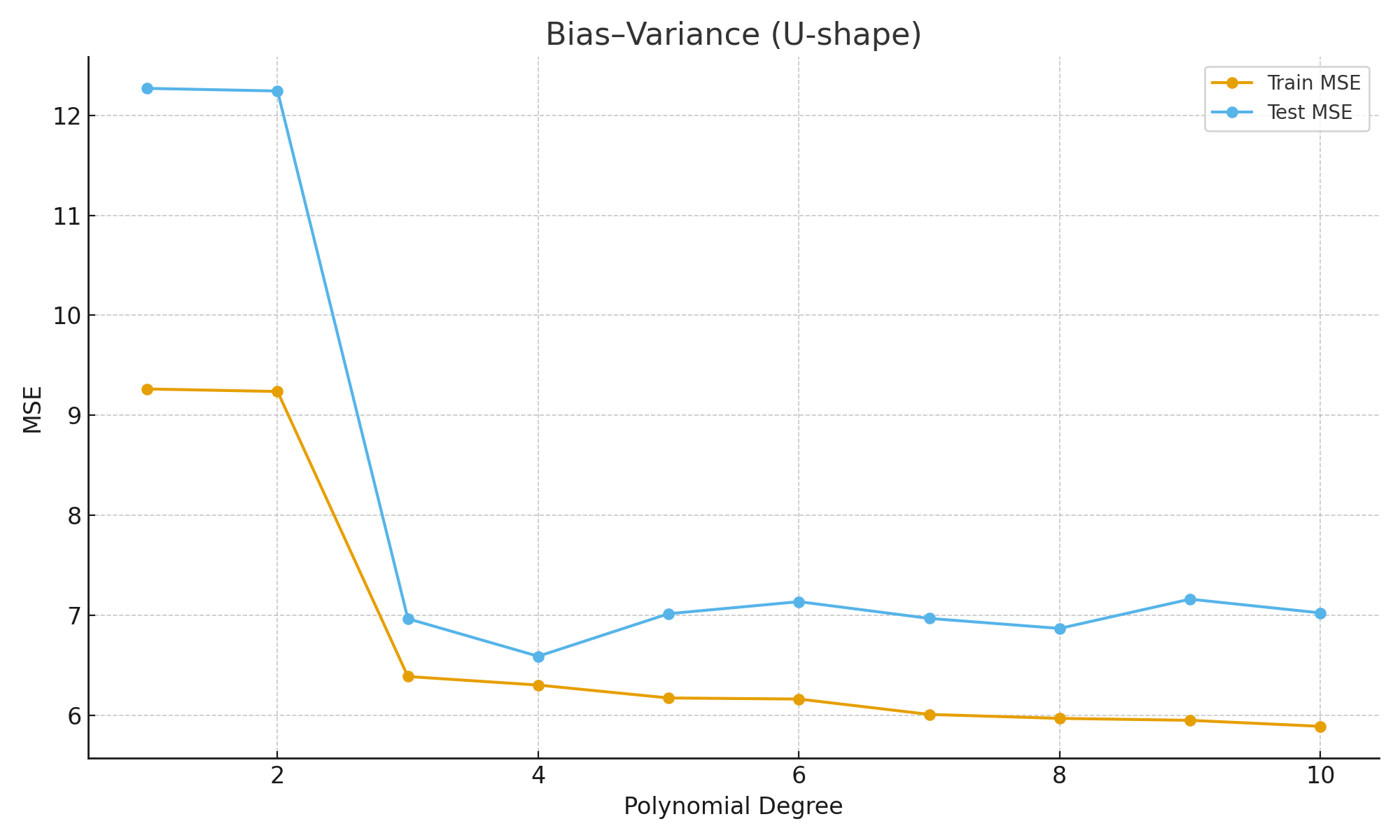
## 4️⃣ (Bổ sung) Bias–Variance Trade-off

### 🔹 Công thức & Ký hiệu

Sai số kỳ vọng trên dữ liệu mới (Expected Test Error) ≈ Bias² + Variance + Noise.  
- Bias: sai lệch do mô hình quá đơn giản (underfitting).  
- Variance: độ nhạy của mô hình với dữ liệu train (mô hình quá phức tạp → overfitting).  
- Noise: nhiễu không thể loại bỏ do dữ liệu.

### Code minh hoạ (dùng dãy MSE theo độ phức tạp)

# Mục đích: tạo train/test MSE theo degree để thấy:  
# - Train MSE giảm dần khi degree tăng (variance tăng)  
# - Test MSE có dạng chữ U: underfitting → vùng tối ưu → overfitting



### Quan sát từ code (Output & Ý nghĩa)

- Degree tối ưu (min Test MSE) ≈ 4  
- Bài học: không chọn degree quá nhỏ (bias cao) hay quá lớn (variance cao); chọn degree tối ưu theo validation/test.

### Mục đích đoạn code

Giúp nhận diện nhanh underfitting/overfitting và định vị "độ phức tạp tối ưu" dựa trên metric trên tập chưa thấy (validation/test).

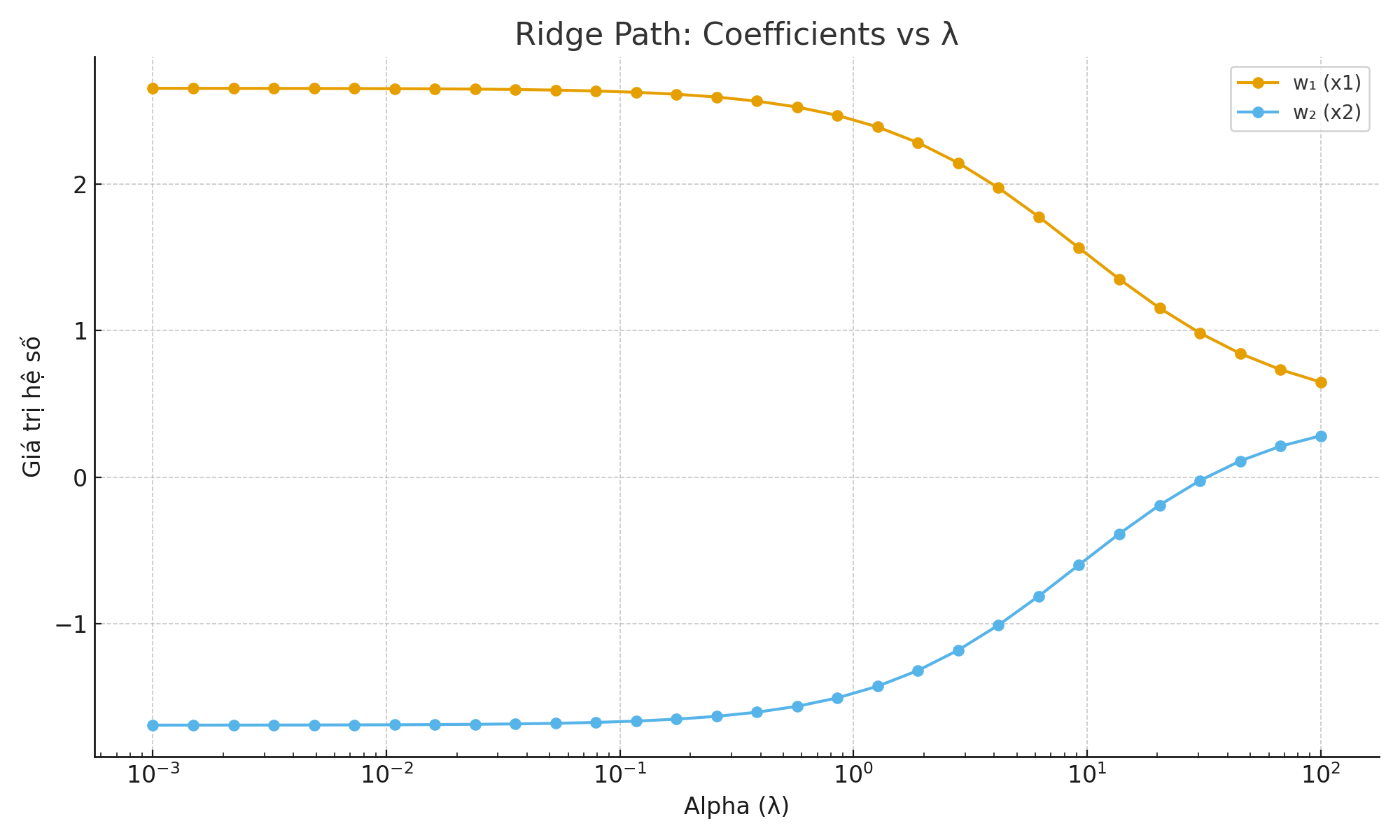
## 5️⃣ (Bổ sung) Regularization & Model Selection

### 🔹 Công thức, Ký hiệu & Tác dụng

- Ridge (L2): minimize Σ(y-ŷ)² + λ Σ wⱼ² → giảm variance, hệ số thu nhỏ mượt.  
- LASSO (L1): minimize Σ(y-ŷ)² + λ Σ |wⱼ| → có thể triệt tiêu hệ số (feature selection).  
- Elastic Net: kết hợp L1 và L2, cân bằng giữa thu nhỏ và chọn lọc.  
Ký hiệu: λ (alpha) là cường độ phạt; wⱼ là hệ số mô hình; intercept thường không bị phạt.

### Code minh hoạ Ridge (đóng kín)

# Mục đích: quan sát "Ridge path" – hệ số biến đổi thế nào khi λ tăng (log-scale)  
# 1) Tạo dữ liệu với hai feature tương quan (collinearity)  
# 2) Fit Ridge với nhiều λ, lưu lại w để vẽ đường cong hệ số  
# 3) Phân tích: λ↑ → |w|↓ → mô hình đơn giản, variance↓



### Output (ví dụ hệ số theo λ)

λ=0.001 → intercept=1.106, w₁=2.651, w₂=-1.690  
λ=0.1 → intercept=1.106, w₁=2.628, w₂=-1.666  
λ=10.0 → intercept=1.119, w₁=1.520, w₂=-0.554

### Ý nghĩa kết quả

- Khi λ tăng, |w₁| và |w₂| giảm → mô hình bớt nhạy với nhiễu/collinearity, tổng quát tốt hơn.  
- Intercept gần như ổn định (không bị phạt).  
- Dùng K-fold CV để chọn λ tối ưu (cân bằng bias–variance).

### Mục đích đoạn code

Minh hoạ vai trò của regularization: kiểm soát độ lớn hệ số để giảm overfitting; đặc biệt hữu ích khi feature tương quan mạnh.

## 6️⃣ (Bổ sung) Summary & Capstone

### Workflow kèm ý nghĩa từng bước

1) EDA → hiểu phân phối/pattern/outlier để chọn mô hình/biến đổi hợp lý.  
2) Feature Engineering → tạo/phối hợp feature có ý nghĩa (scale, polynomial,...).  
3) Split/CV → đánh giá công bằng, chọn tham số/độ phức tạp (degree, λ).  
4) Train → huấn luyện mô hình tối thiểu hoá loss.  
5) Evaluate → đo bằng MAE/MSE/RMSE/R² trên tập chưa thấy.  
6) Tune → Grid/Random search, chọn cấu hình tốt.  
7) Deploy → đóng gói/monitor, cập nhật khi drift.