# Fondamenti di Analisi dei Dati

from data analysis to predictive techniques

Prof. Antonino Furnari (antonino.furnari@unict.it)

Corso di Studi in Informatica
Dip. di Matematica e Informatica
Università di Catania

Università di Catania

# Beyond Linear Regression

Extending the linear model to handle complex, non-linear relationships while managing the critical trade-off between bias and variance in predictive modeling.

# The Journey Ahead

01

### Statistical Fixes

Extending the linear model with interaction terms and polynomial features

02

### Quadratic & Polynomial Regression

Capturing non-linear patterns in data through higher-degree terms

03

### Machine Learning Perspective

Shifting focus from understanding to prediction accuracy

04

### Overfitting Problem

Understanding when models become too powerful

05

### Regularization Solutions

Ridge and Lasso techniques to control model complexity

06

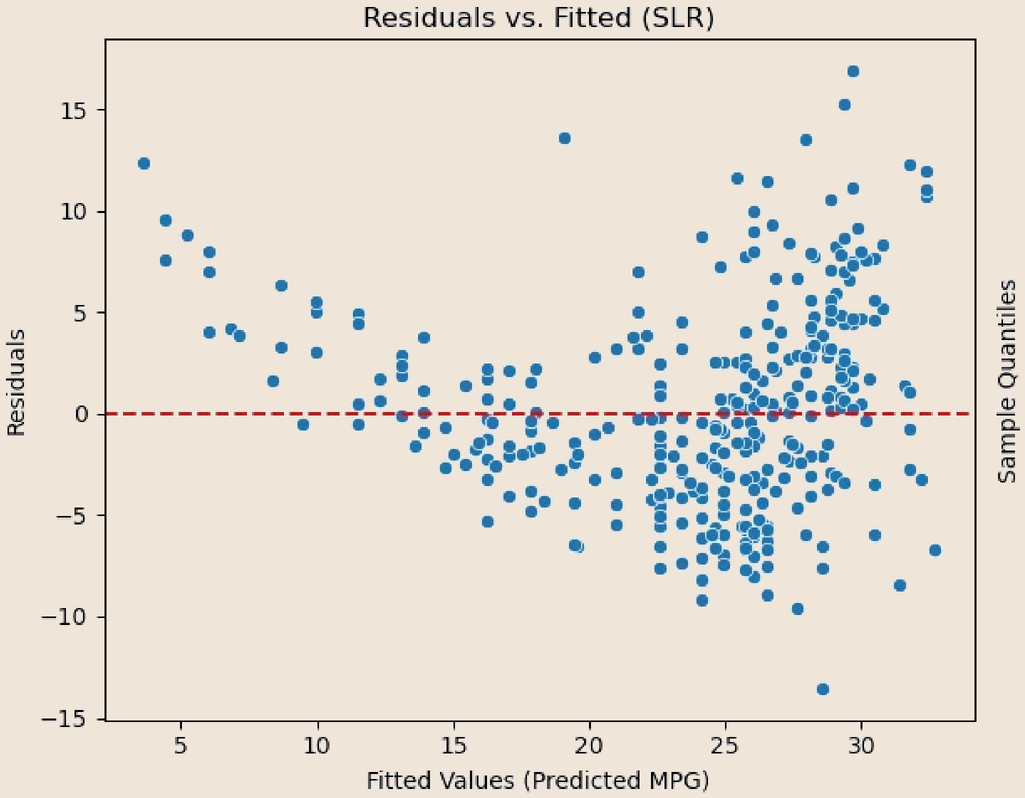
### Practical Implementation

The scikit-learn workflow for real-world applications

# Problems with Simple Linear Models

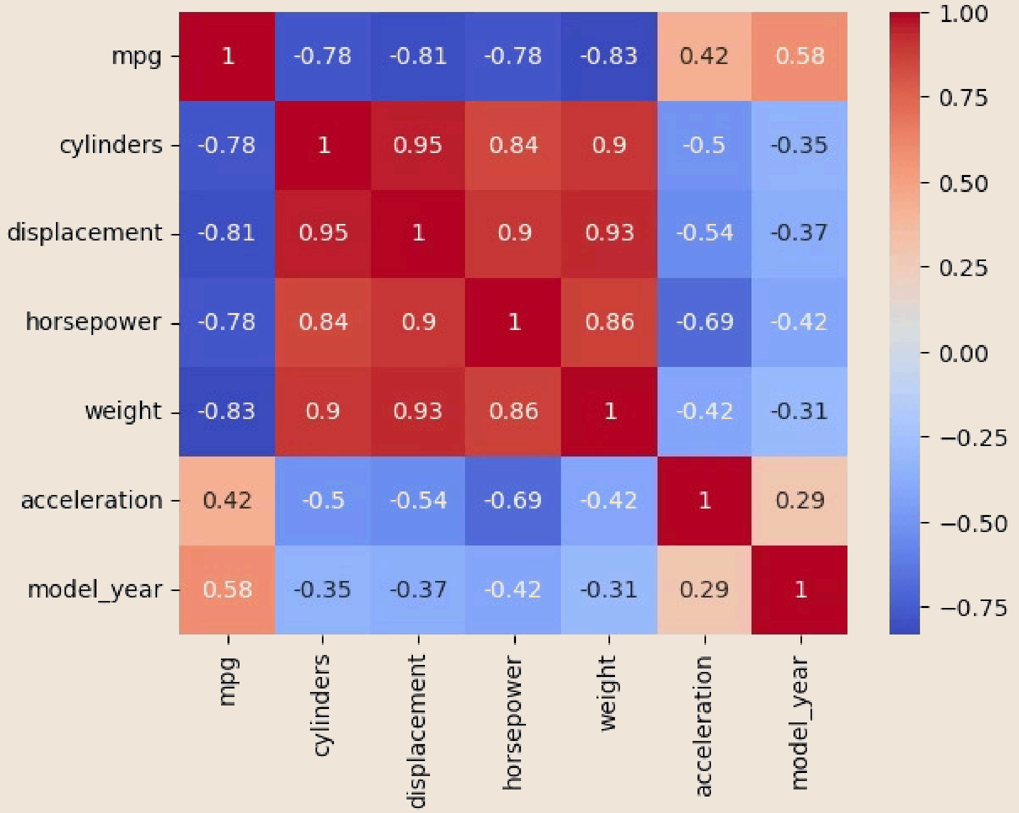
## 1. Non-Linear Patterns

When modeling mpg ~ horsepower, residual plots revealed a clear U-shaped pattern. This indicates **underfitting** (high bias) because the real-world relationship isn't linear.



## 2. Multicollinearity

Using multiple correlated predictors like horsepower and weight caused unstable p-values and confused interpretation. High correlation between predictors makes coefficient estimates unreliable.



| | mpg | cylinders | displacement | horsepower | weight | acceleration | model_year |
|---|---|---|---|---|---|---|---|
| mpg | 1 | -0.78 | -0.81 | -0.78 | -0.83 | 0.42 | 0.58 |
| cylinders | -0.78 | 1 | 0.95 | 0.84 | 0.9 | -0.5 | -0.35 |
| displacement | -0.81 | 0.95 | 1 | 0.9 | 0.93 | -0.54 | -0.37 |
| horsepower | -0.78 | 0.84 | 0.9 | 1 | 0.86 | -0.69 | -0.42 |
| weight | -0.83 | 0.9 | 0.93 | 0.86 | 1 | -0.42 | -0.31 |
| acceleration | 0.42 | -0.5 | -0.54 | -0.69 | -0.42 | 1 | 0.29 |
| model_year | 0.58 | -0.35 | -0.37 | -0.42 | -0.31 | 0.29 | 1 |

Today we'll fix these problems by extending the linear model, though we'll sacrifice some interpretability in exchange for predictive power.

# Interaction Terms: When Effects Depend on Context

Simple linear models assume additive effects. But what if the effect of horsepower on MPG depends on the car's weight?

## Additive Model

$$mpg = \beta_0 + \beta_1 \text{horsepower} + \beta_2 \text{weight}$$

Effect of horsepower is constant: $\beta_1$

**Adj. R-squared:** 0.7049

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     39.9359      0.717     55.660      0.000      38.525      41.347
horsepower    -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
```

## Interaction Model

$$mpg = \beta_0 + \beta_1 \text{horsepower} + \beta_2 \text{weight} + \beta_3 (\text{horsepower} \times \text{weight})$$

Effect varies: $\beta_1 + \beta_3 \times \text{weight}$

**Adj. R-squared:** 0.7465 ✓

```
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            63.5579      2.343     27.127      0.000      58.951      68.164
horsepower           -0.2508      0.027     -9.195      0.000      -0.304      -0.197
weight               -0.0108      0.001    -13.921      0.000      -0.012      -0.009
horsepower:weight  5.355e-05   6.65e-06      8.054      0.000    4.05e-05    6.66e-05
=====================================================================================
```

## Interpretation of the Interaction Model

How to interpret the model? We can rewrite it as:

$$mpg = \beta_0 + (\beta_1 + \beta_3 \times \text{weight}) \times \text{horsepower} + \beta_2 \text{weight}$$

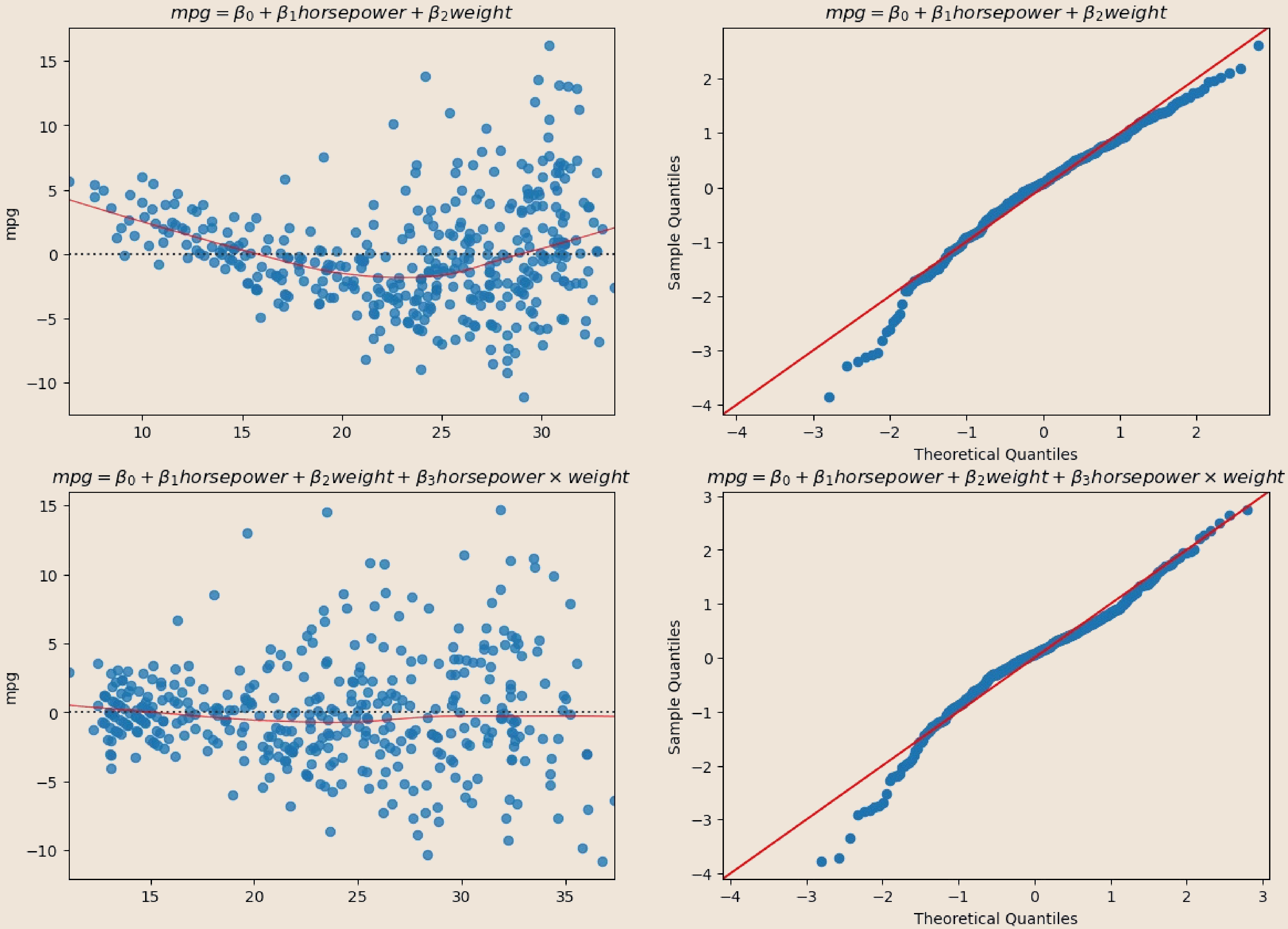
The effect of horsepower is no longer a single number, $\beta_1$. It is now $(\beta_1 + \beta_3 \times \text{weight})$. This means that the effect of horsepower **depends one** the weight of the car.

In practice, **when we increase weight by one unit, we increase the effect of horsepower on mpg by $\beta_3$ units.** This means that for heavier cars, horsepower has a less negative effect on mpg.

This makes sense mechanically: in a very heavy car, horsepower is needed just to move it, so the efficiency penalty is smaller compared to a light car where extra horsepower is “overkill.

# Comparing the new Model to a Simple Regressor

The model is a little better!

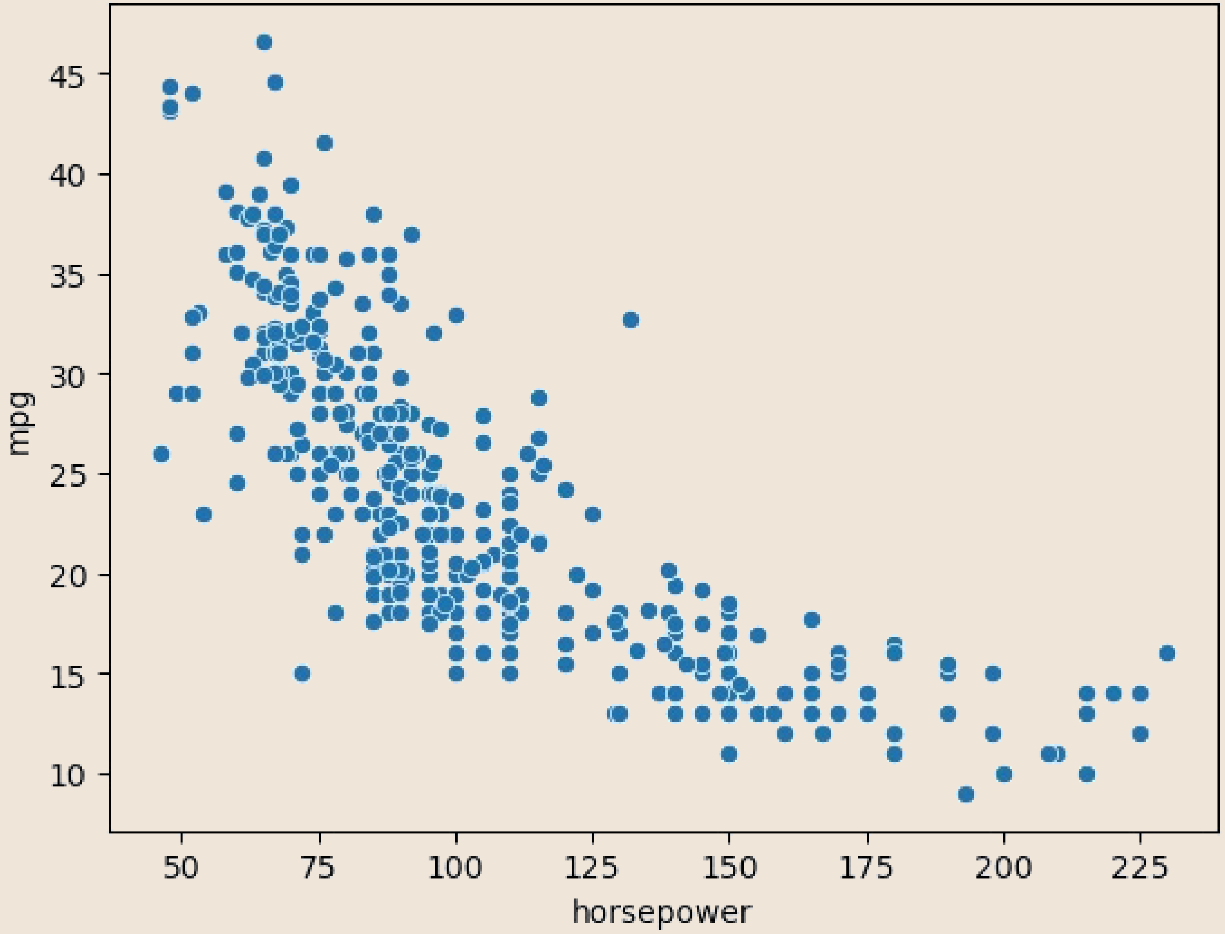


# The Non-Linear Reality

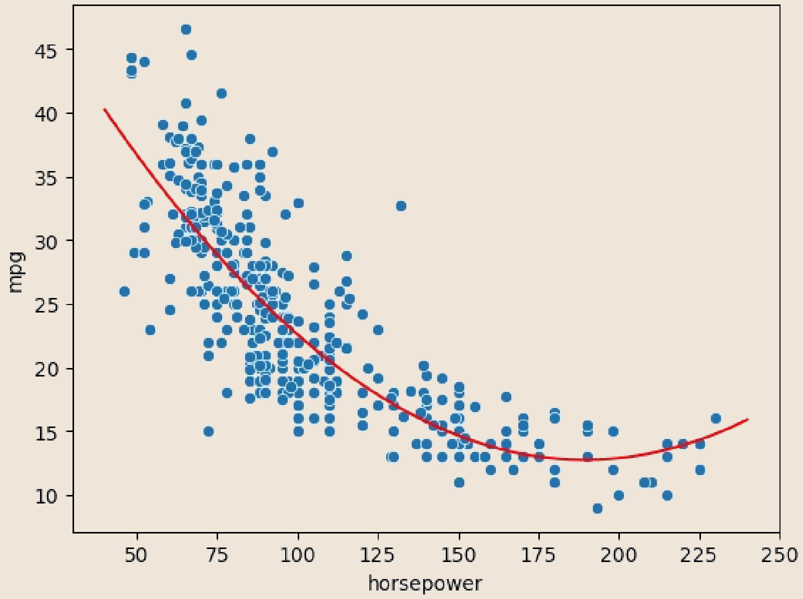
## The Problem

The relationship between MPG and horsepower isn't linear—it's curved. A straight line simply can't capture this pattern.

We need a model that can bend and curve to fit the data's natural shape.



# Quadratic Regression: Adding Curvature



We can capture the curved by adding a squared term:

$$mpg = \beta_0 + \beta_1 \text{horsepower} + \beta_2 \text{horsepower}^2$$

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 56.9001 | 1.800 | 31.604 | 0.000 | 53.360 | 60.440 |
| **horsepower** | -0.4662 | 0.031 | -14.978 | 0.000 | -0.527 | -0.405 |
| **I(horsepower ** 2)** | 0.0012 | 0.000 | 10.080 | 0.000 | 0.001 | 0.001 |

This model has a higher $R^2$ of $0.688$, larger than the base $0.608$.

We now have the model:

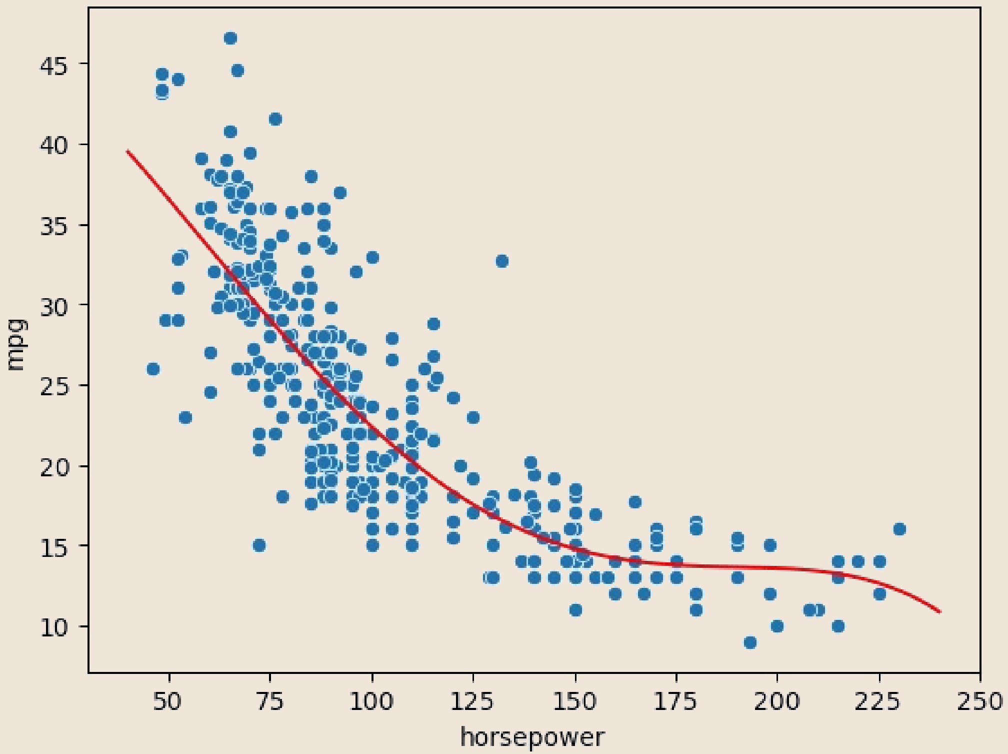
$$y = 58.9001 - 0.4662x + 0.0012x^2$$

While we have a better model, we lost interpretability. How do we interpret the coefficient of the quadratic term? How does the interpretation of other coefficients change?

# Polynomial Regression: More Flexibility

We can extend this idea to higher-degree polynomials. For degree *d*=4:

$$mpg = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$



## Key Insight

While the polynomial function is non-linear in the variable, it's still **linear in its coefficients**. This means we can solve it using ordinary least squares!

We simply create new variables for $x^2$, $x^3$, etc., and treat it as a linear regression problem.

For instance, this model:

$$y = \beta_0 + \beta_1 x + \beta_2 y$$

becomes in the case of degree $d = 2$:

$$y = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy$$

## The Interpretability Trade-Off

# What is the "effect" of horsepower on MPG?

### Linear Model

Simple answer: $\beta_1$

One number tells the whole story. Easy to explain and understand.

### Quadratic Model

The effect *depends on the value of horsepower*. No single number captures it.

---

This is the pivot point. If we're willing to sacrifice simple interpretability for better fit, why stop here? This is where we cross from **statistical inference** into **Machine Learning**.

# Shifting Perspectives: From Understanding to Prediction

### Statistical Approach

- Focus on p-values and $R^2$
- Interpret coefficients
- Understand relationships
- Use training data metrics

### Machine Learning Approach

- Focus on predictive accuracy
- Optimize for unseen data
- Embrace complexity
- Use test set metrics

When prediction is the goal, we only care about performance on **new, unseen data** (the test set). This requires new metrics and a new mindset.

# Metrics for Prediction

## 1

### Mean Squared Error (MSE)

$$MSE_{test} = \frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2$$

Average of squared errors. Problem: units are squared (e.g., meters²), making interpretation difficult.

## 2

### Root Mean Squared Error (RMSE)

$$RMSE_{test} = \sqrt{MSE_{test}}$$

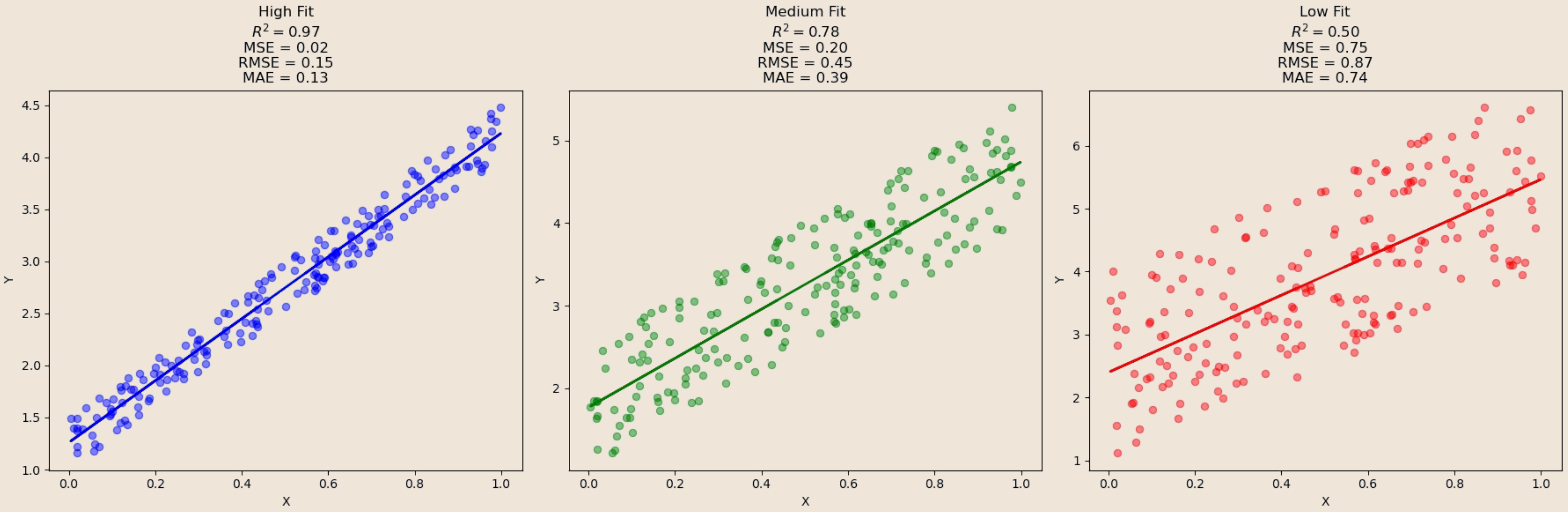
**Most common metric.** Same units as Y. Penalizes large errors more heavily. Interpretation: "On average, predictions are wrong by [RMSE value]."

## 3

### Mean Absolute Error (MAE)

$$MAE_{test} = \frac{1}{m}\sum_{j=1}^{m}|y_j - \hat{y}_j|$$

Average absolute error. Same units as Y. More robust to outliers than RMSE—doesn't disproportionately penalize large errors.



High Fit
$R^2 = 0.97$
MSE = 0.02
RMSE = 0.15
MAE = 0.13

Medium Fit
$R^2 = 0.78$
MSE = 0.20
RMSE = 0.45
MAE = 0.39

Low Fit
$R^2 = 0.50$
MSE = 0.75
RMSE = 0.87
MAE = 0.74

## The Overfitting Danger

# If degree=2 is good, is degree=20 better?

### Underfitting

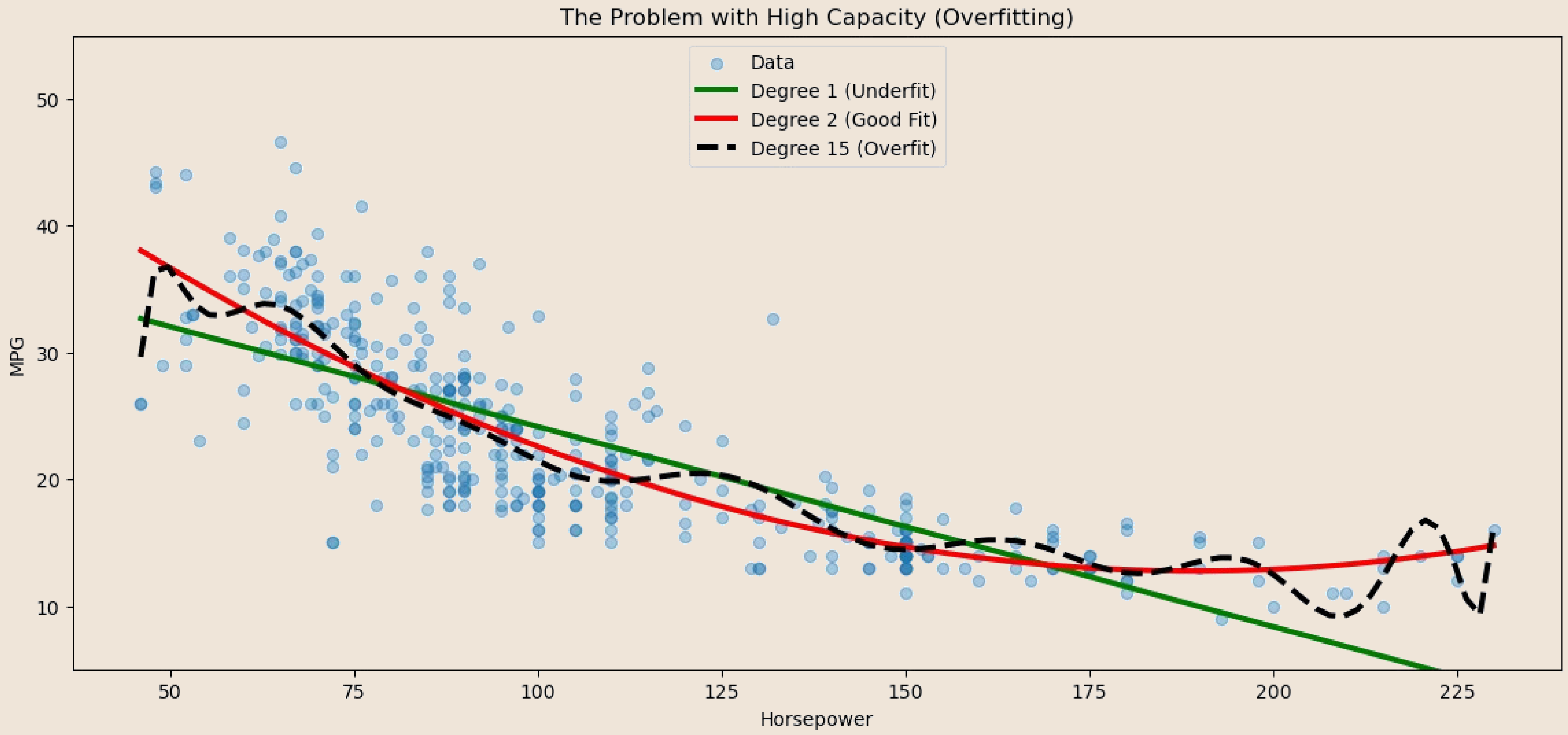
Linear model is too simple. Misses the pattern. **High bias.**

### Just Right

Quadratic model captures the signal. Balanced bias-variance.

### Overfitting

High-degree polynomial learns the noise. **High variance.**



A high-degree polynomial can "wiggle" to fit every single training point, achieving perfect R² on training data but failing miserably on new data.

# The Solution: Regularization

Regularization adds a penalty to the cost function to discourage model complexity:

$$RSS + \text{Penalty Term}$$

1

### RSS Term

Pushes model to fit the data (reduces bias)

2

### Penalty Term

Forces coefficients to stay small (reduces variance)

3

### Balance

Find the sweet spot between bias and variance

A "wiggly" overfit curve requires massive β coefficients. By penalizing large coefficients, we force the model to find a simpler, smoother curve.

**Bonus:** Regularization also solves multicollinearity by stabilizing the coefficient estimates!

# Ridge Regression ($L_2$Penalty)

Ridge adds a penalty proportional to the **sum of squares** of coefficients:

$$RSS_{L2} = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

This is solved with OLS as follows:

$$\hat{\beta}_{Ridge} = (X^T X + \lambda I)^{-1} X^T y$$

where the $\lambda I$ term improves conditioning (invertibility) of the matrix even in the presence of multicollinearity.

### λ = 0

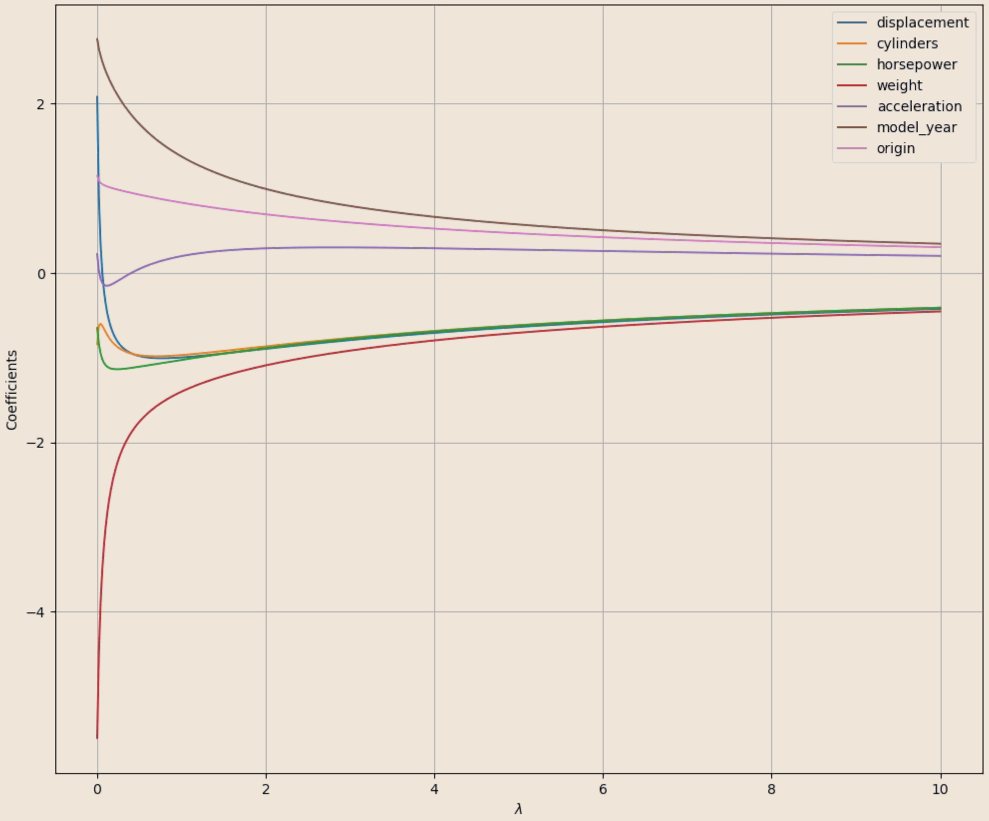
Standard OLS. No penalty. Risk of overfitting.

### λ = optimal

Sweet spot. Balanced bias-variance trade-off.

### λ → ∞

All coefficients forced to zero. Flat line. Underfitting.



**Critical:** Standardize (z-score) all features first! Otherwise, features with different scales will be penalized unfairly.

# Interpreting Ridge Coefficients

| Variable | Ridge | OLS |
|---|---|---|
| displacement | -0.998 | 2.079 |
| cylinders | -0.970 | -0.841 |
| horsepower | -1.027 | -0.652 |
| weight | -1.402 | -5.492 |
| acceleration | 0.200 | 0.222 |
| model_year | 1.376 | 2.762 |
| origin | 0.830 | 1.147 |

Ridge parameters have smaller magnitude due to regularization. They can't be interpreted statistically like OLS coefficients. Instead, they indicate **relative importance** of variables for prediction.

# Lasso Regression ($L_1$ Penalty)

Lasso uses the **sum of absolute values** of coefficients:

$$RSS_{L1} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j|$$

## Ridge vs. Lasso

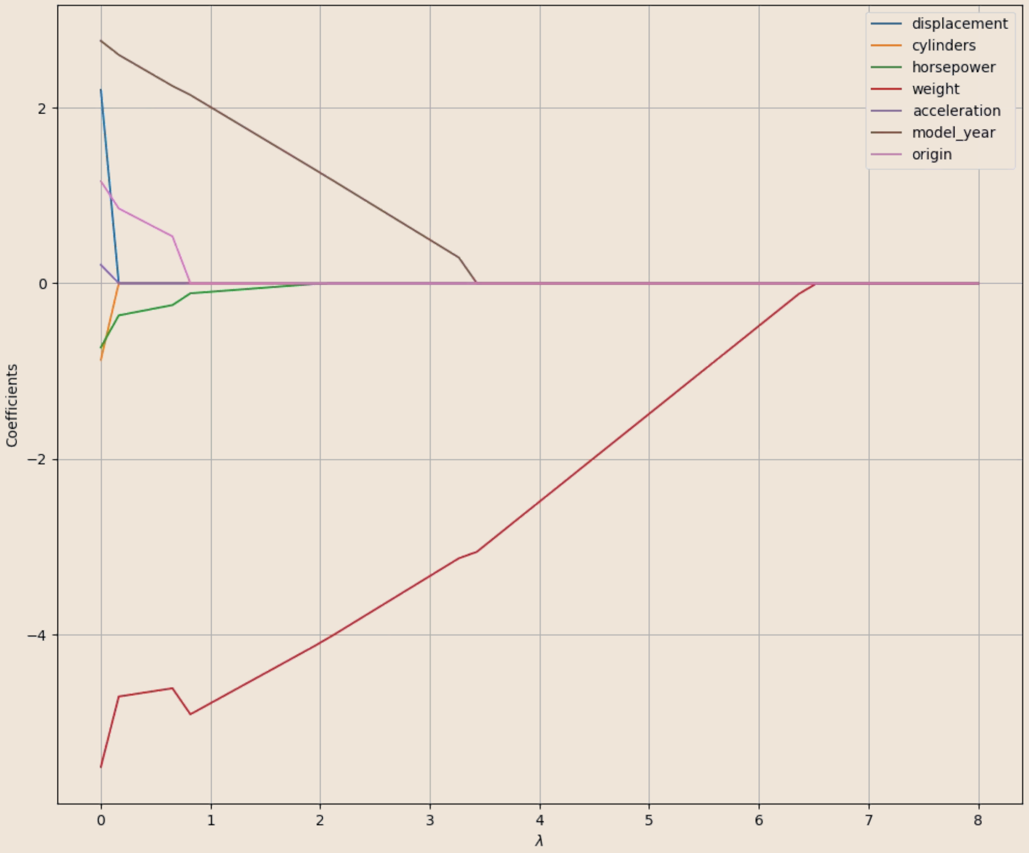
- **Ridge:** Shrinks coefficients smoothly
- **Lasso:** Sets coefficients exactly to zero

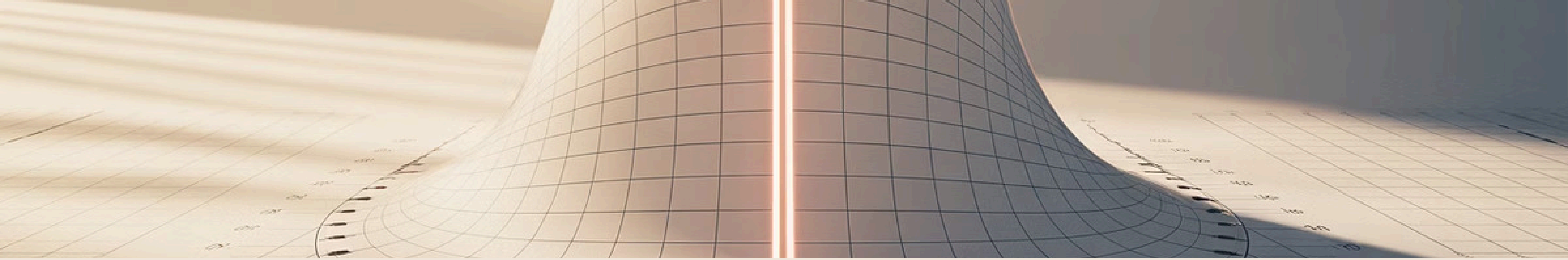
The "sharp corners" of the absolute value function allow Lasso to eliminate features entirely.

## Lasso's Characteristic

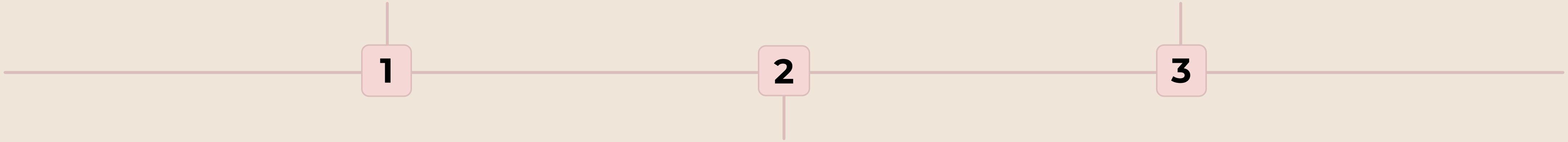
**Automatic feature selection!**

When you have hundreds of features, **Lasso identifies which ones matter and removes the rest.** This creates simpler, more interpretable models.





# The Bias-Variance Trade-Off



### λ = 0 (No Penalty)

Low Bias, High Variance

Overfitting: perfect fit on training data, poor on test data

### λ = Optimal

Balanced

Sweet spot: lowest total error on test data

### λ → ∞ (Heavy Penalty)

High Bias, Low Variance

Underfitting: flat line, poor fit everywhere

λ is our "tradeoff knob"—the entire goal of machine learning is finding the optimal λ that minimizes test error.

# Validation Curve: Finding the Sweet Spot

### Left Side (λ ≈ 0.001)

Training error very low, test error high. **Overfitting** (high variance region).

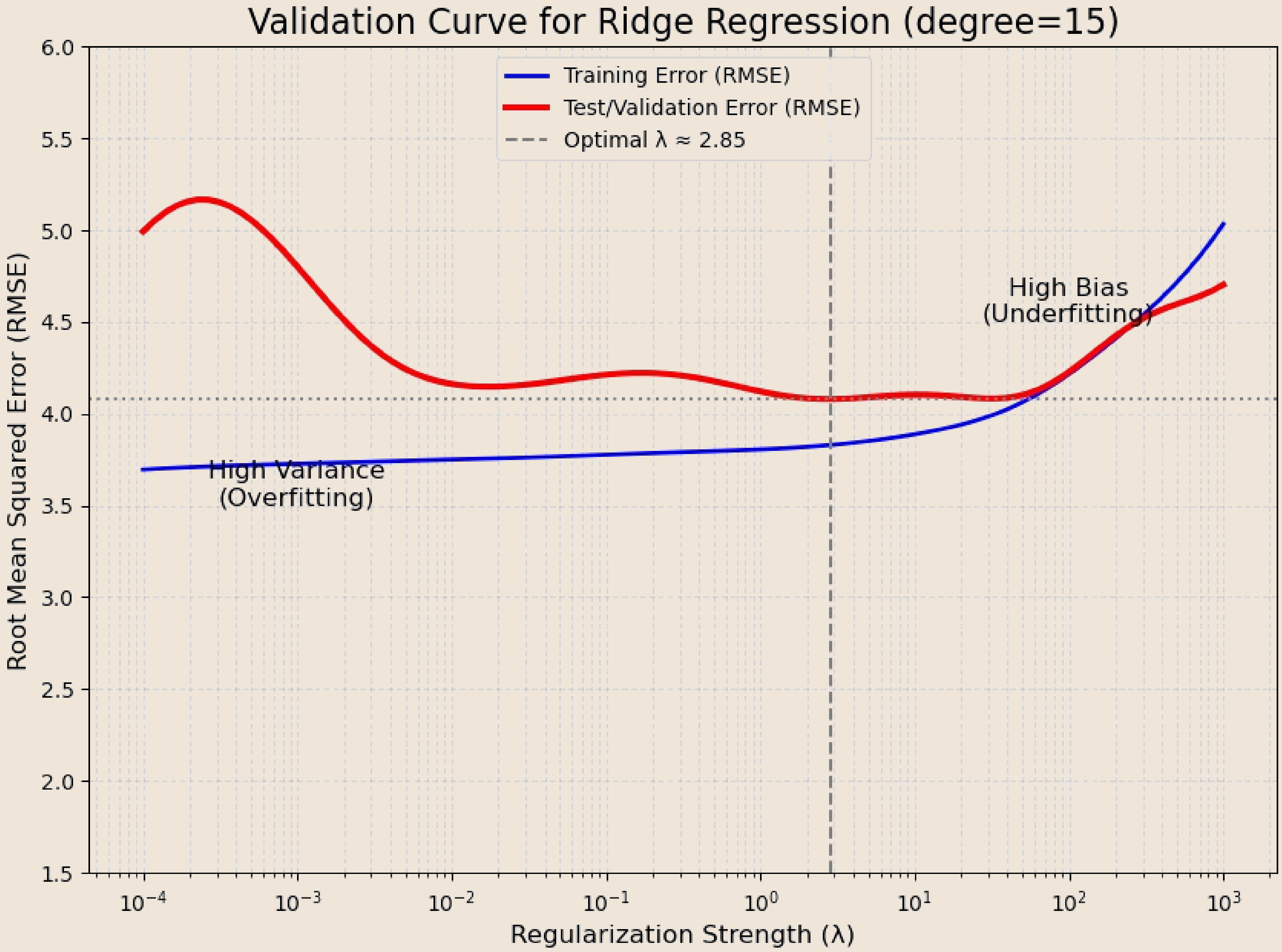
### Sweet Spot (Dashed Line)

Test error at minimum. **Optimal λ**—perfect balance between bias and variance.

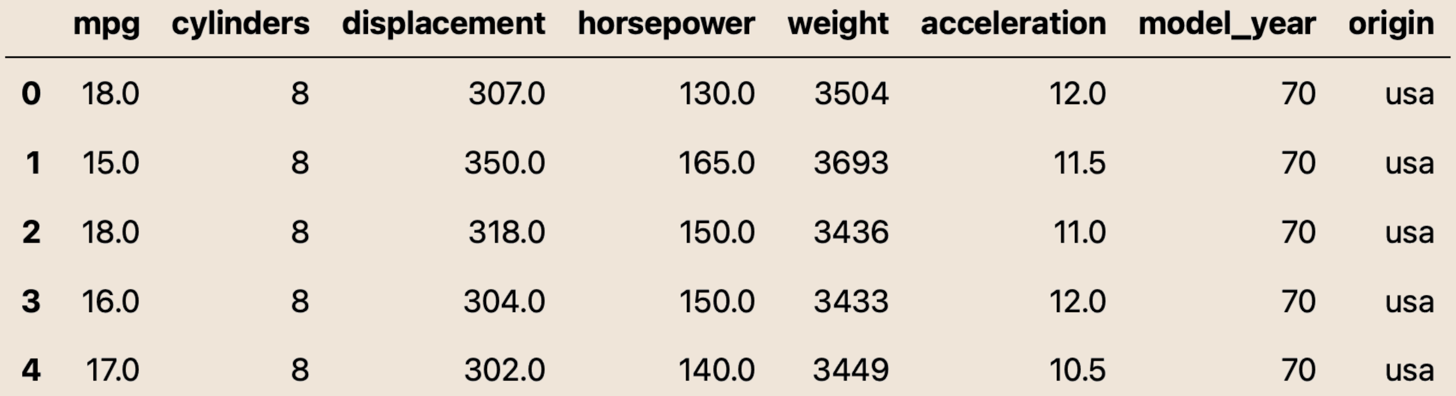
### Right Side (λ > 1000)

Both errors high and similar. **Underfitting** (high bias region).

This visualization shows how regularization controls the bias-variance trade-off for a degree-15 polynomial. Cross-validation helps us find the optimal λ systematically.



# Linear Regression in Python



| | mpg | cylinders | displacement | horsepower | weight | acceleration | model_year | origin |
|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | usa |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | usa |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | usa |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | usa |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | usa |

# Conclusions and Next Steps

## We Have Explored:

- Interaction Terms
- Quadratic Model
- Polynomial Model
- Regularization

In the next lectures, we will look at classification problems

### References

- Chapter 3 of [1]
- Parts of chapter 11 of [2]

[1] Heumann, Christian, and Michael Schomaker Shalabh. Introduction to statistics and data analysis. Springer International Publishing Switzerland, 2016.

[2] James, Gareth Gareth Michael. An introduction to statistical learning: with applications in Python, 2023.https://www.statlearning.com