CAP 4630 – Linear Regression II

Instructor: Aakash Kumar

University of Central Florida

Multiple Linear Regression

- Simple Linear Regression:
  - Involves **one independent variable** and **one dependent variable**.
  - Example Equation: $y = \theta_0 + \theta_1 x_1$

    - y: Dependent variable (response)
    - x_1: Independent variable (predictor)
    - θ_0: Intercept
    - θ_1: Slope

- Multiple Linear Regression:
  - Used when there are **multiple independent variables**.
  - Example Equation: $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

    - $x_1, x_2, \ldots, x_n$: Independent variables (predictors)
    - $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$: Coefficients of the predictors

Dataset for Multiple Linear Regression

- In multiple linear regression, we often work with a **dataset** that includes multiple predictor (independent) variables.
- The dataset is organized in a **tabular form**, where:
  - **Rows**: Each row represents one observation (or data point).
  - **Columns**: The one column represents the dependent variable (the outcome we want to predict), and other columns represent independent variables (the predictors).

Square Feet	Bedrooms	Bathrooms	Age (Years)	Location (Distance from City Center in miles)	Price (Y)
2100	3	2	10	5.5	$450,000
1600	2	2	20	10.2	$320,000
2500	4	3	5	7.8	$500,000
1800	3	2	15	6.1	$380,000
3000	5	4	2	3.2	$670,000

Understanding the Cost Function in Linear Regression

- **The cost function (also known as the loss function)** quantifies the error between the predicted values and the actual values.
- In linear regression, we use the **Mean Squared Error (MSE)** cost function to measure the accuracy of the predictions.
- Mean Squared Error (MSE) Formula:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Where:

  - m is the number of training examples.
  - $h_\theta(x^{(i)})$ is the predicted value from the hypothesis.
  - $y^{(i)}$ is the actual value.

Gradient Descent

- **Purpose**: Gradient descent is an iterative optimization algorithm used to minimize the cost function **J(θ)** by updating the model parameters **θ**.
- **Key Idea:** Gradient descent moves the parameters in the direction of the negative gradient of the cost function to reduce error.
- **Gradient Descent Update Rule:** For each parameter **θj**

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

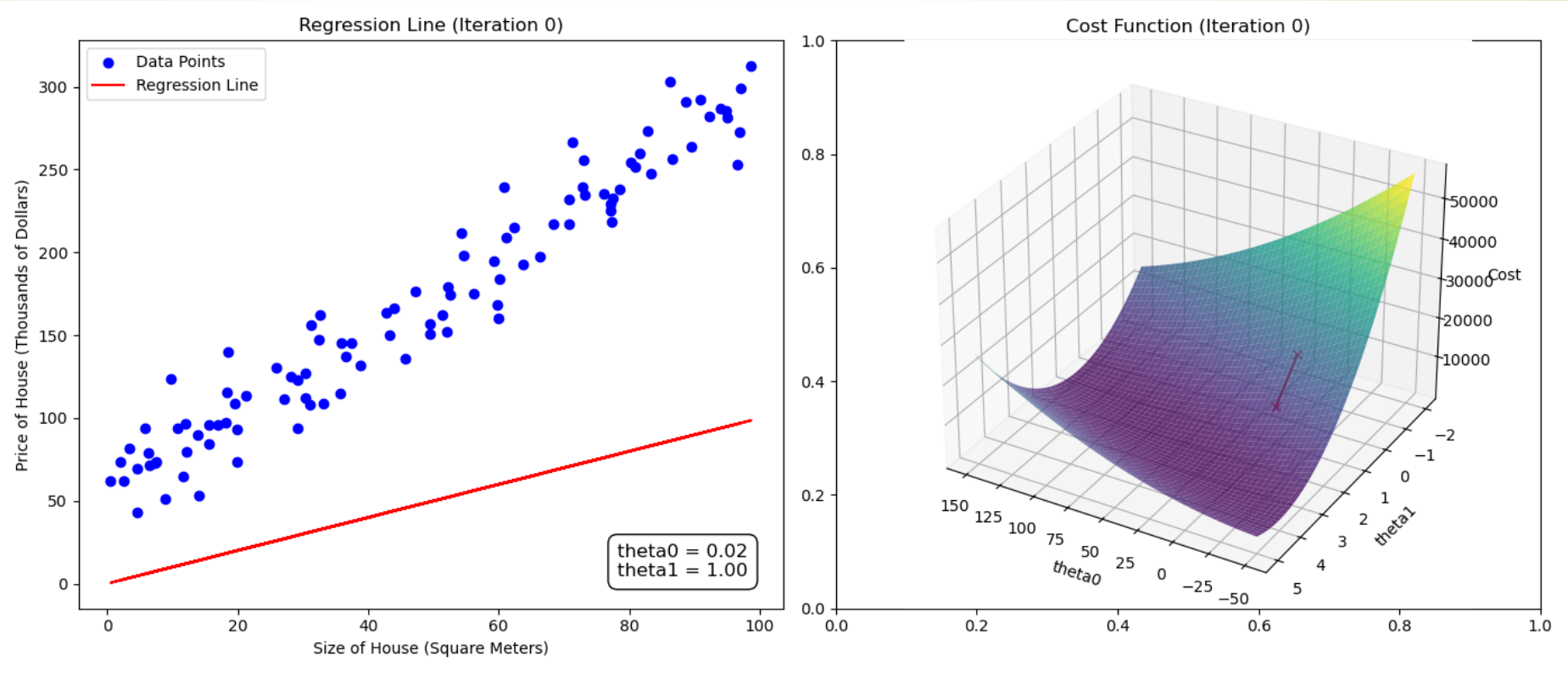
- **Iterative Process:**
  - Start with random initial values for **θ**.
  - Compute the cost function **J(θ)**.
  - Update θ using the update rule.
  - Repeat until the cost function converges (i.e., stops changing significantly).

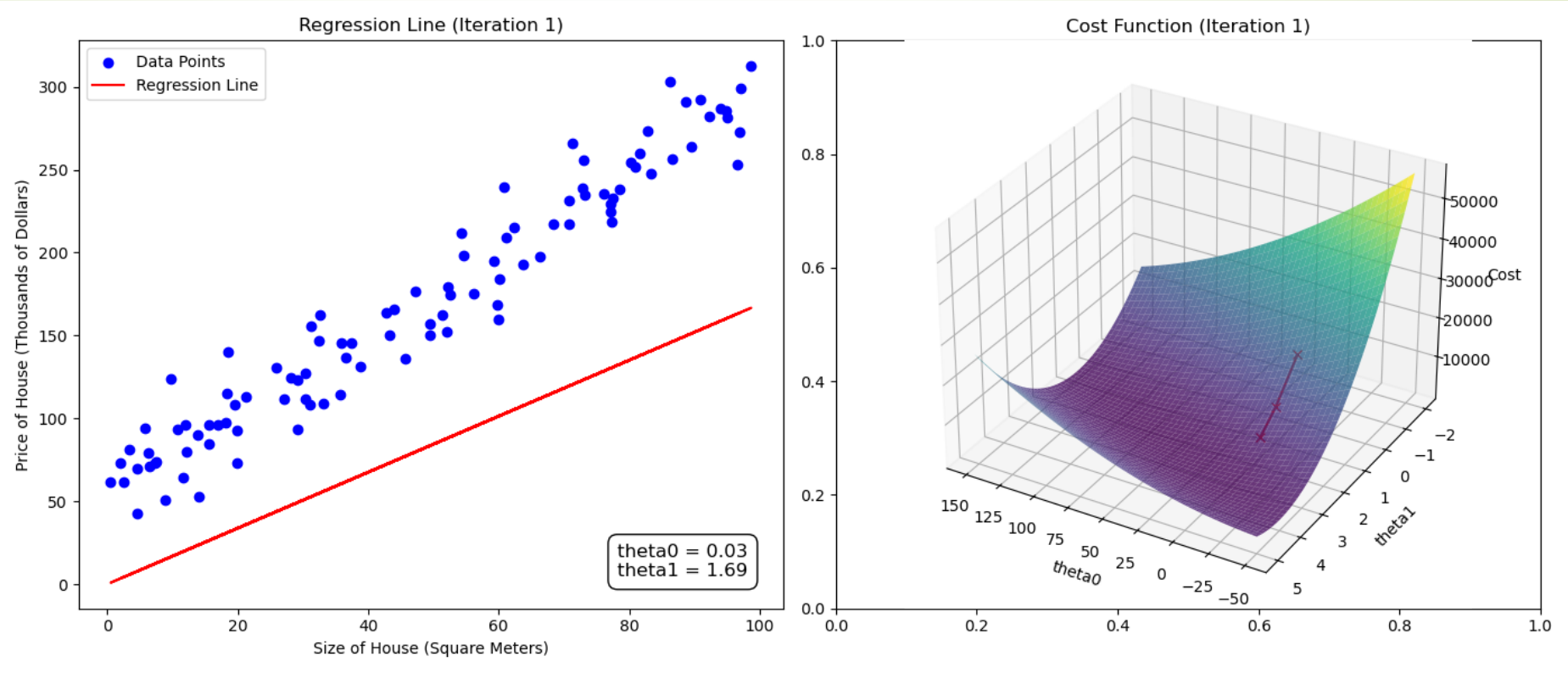
Learning Rate in Gradient Descent

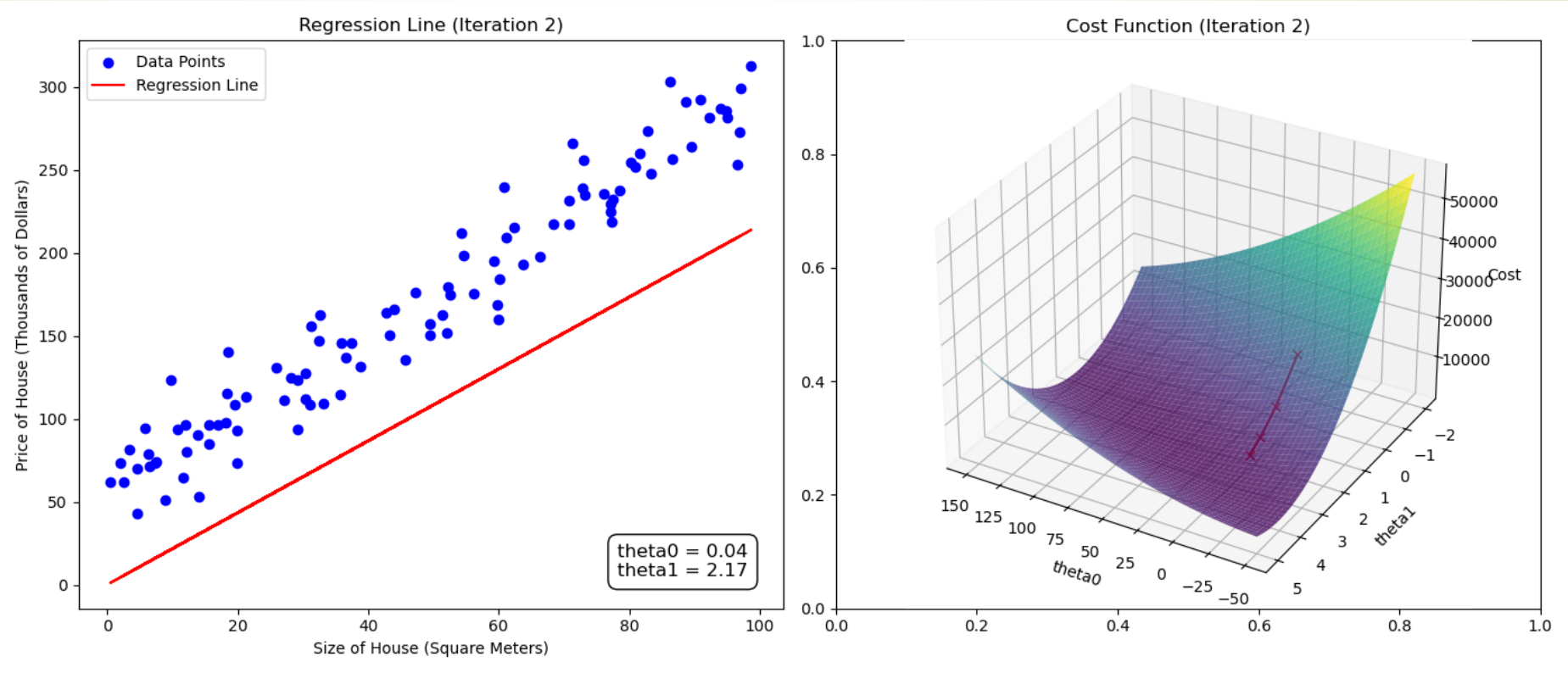
- Learning Rate : This is a hyperparameter that determines the step size during the parameter updates in gradient descent.

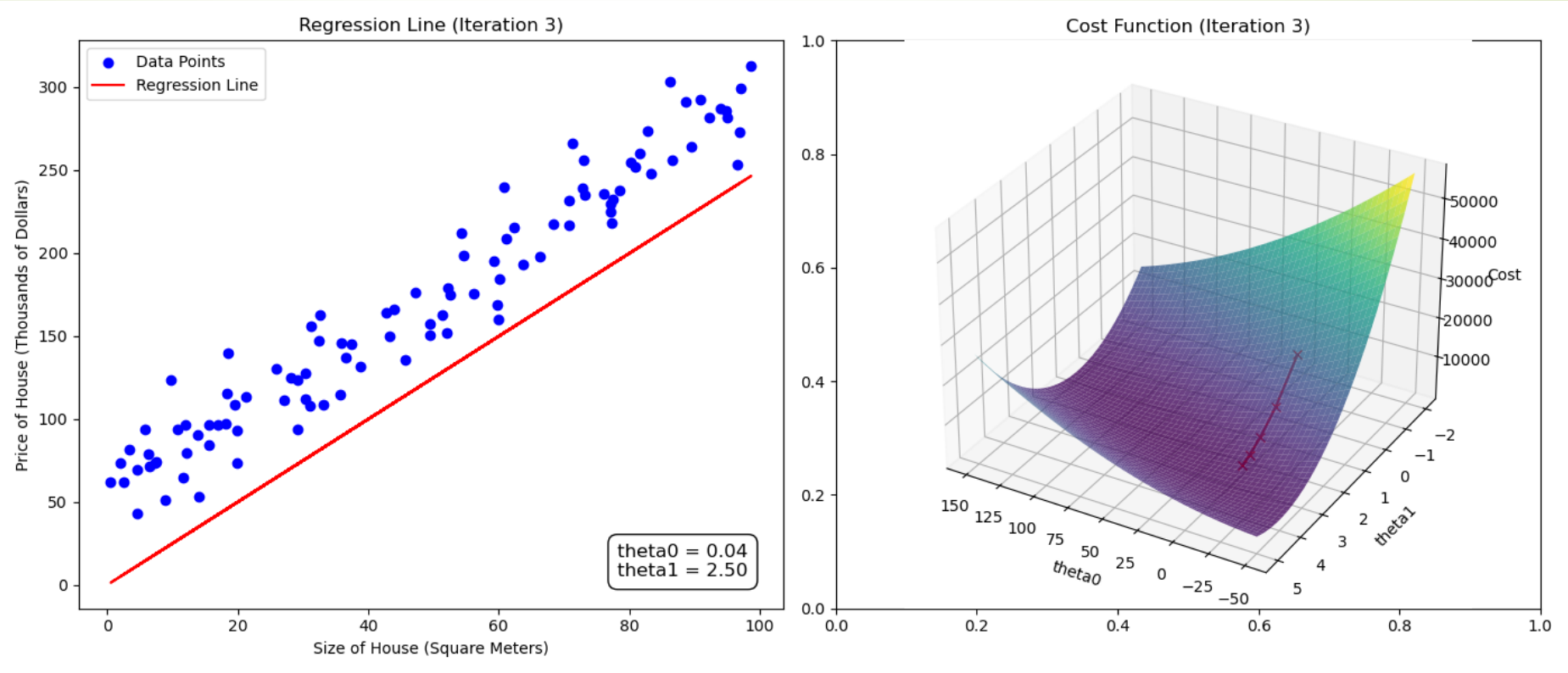
- Choosing the Right Learning Rate:
  - Too Small: If the learning rate is too small, gradient descent will take many iterations to converge, making the process slow.
  - Too Large: If the learning rate is too large, gradient descent may overshoot the minimum or even fail to converge, causing the algorithm to be unstable.

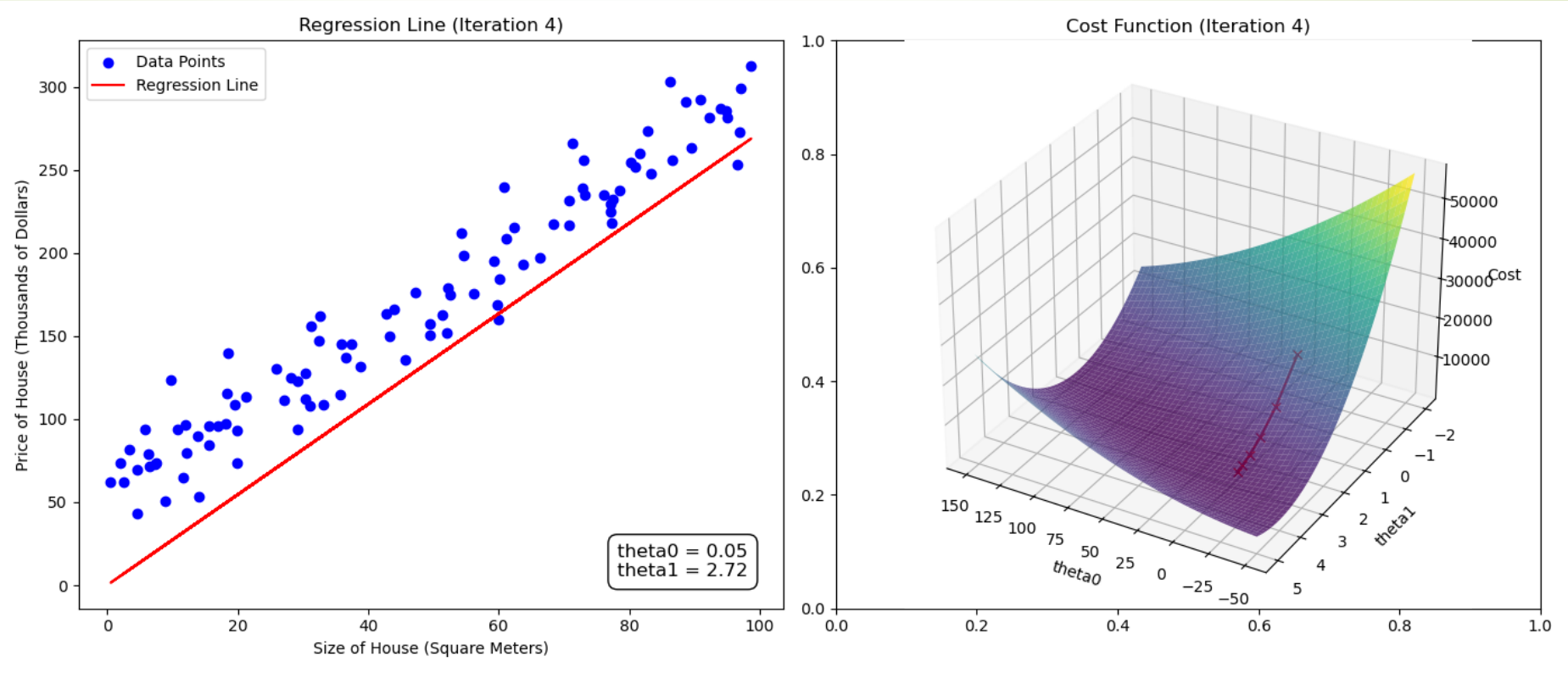
- Effect of Learning Rate:
  - Small Learning Rate: Slow convergence, but guarantees a smooth decrease in the cost function.
  - Large Learning Rate: Faster convergence, but may result in overshooting the optimal parameters and oscillating around the minimum.

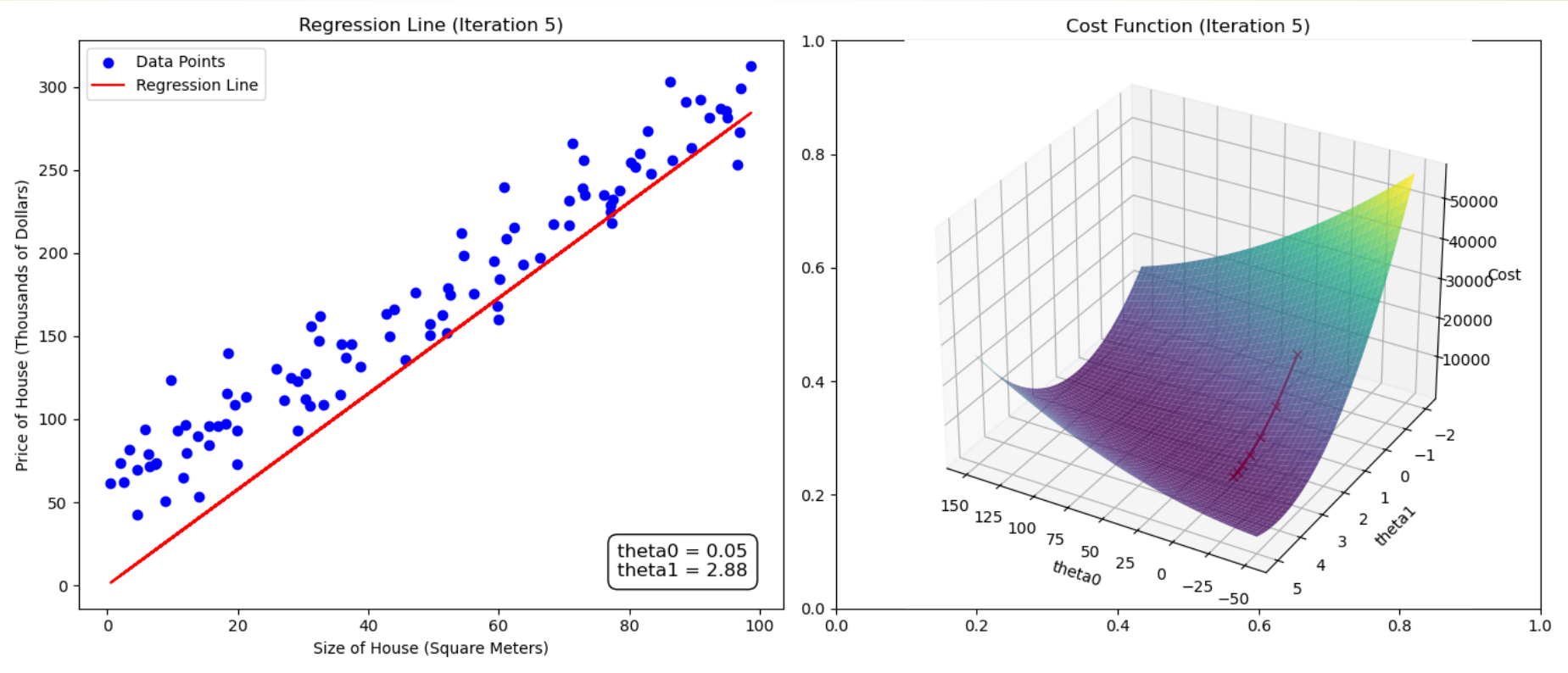

Regression Line (Iteration 0)
Data Points
Regression Line
theta0 = 0.02
theta1 = 1.00
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 0)
Cost
theta0
theta1

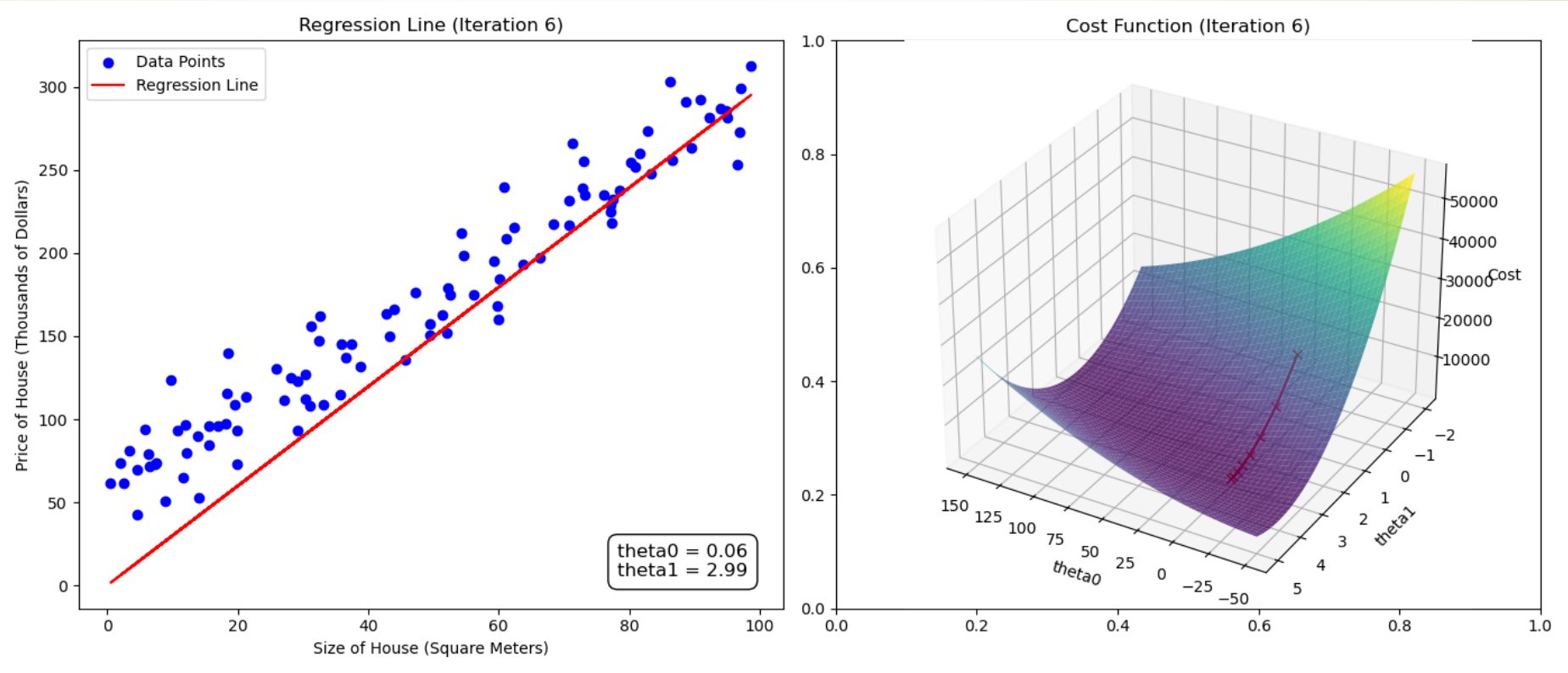

Regression Line (Iteration 1)
Data Points
Regression Line
theta0 = 0.03
theta1 = 1.69
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 1)
Cost
theta0
theta1

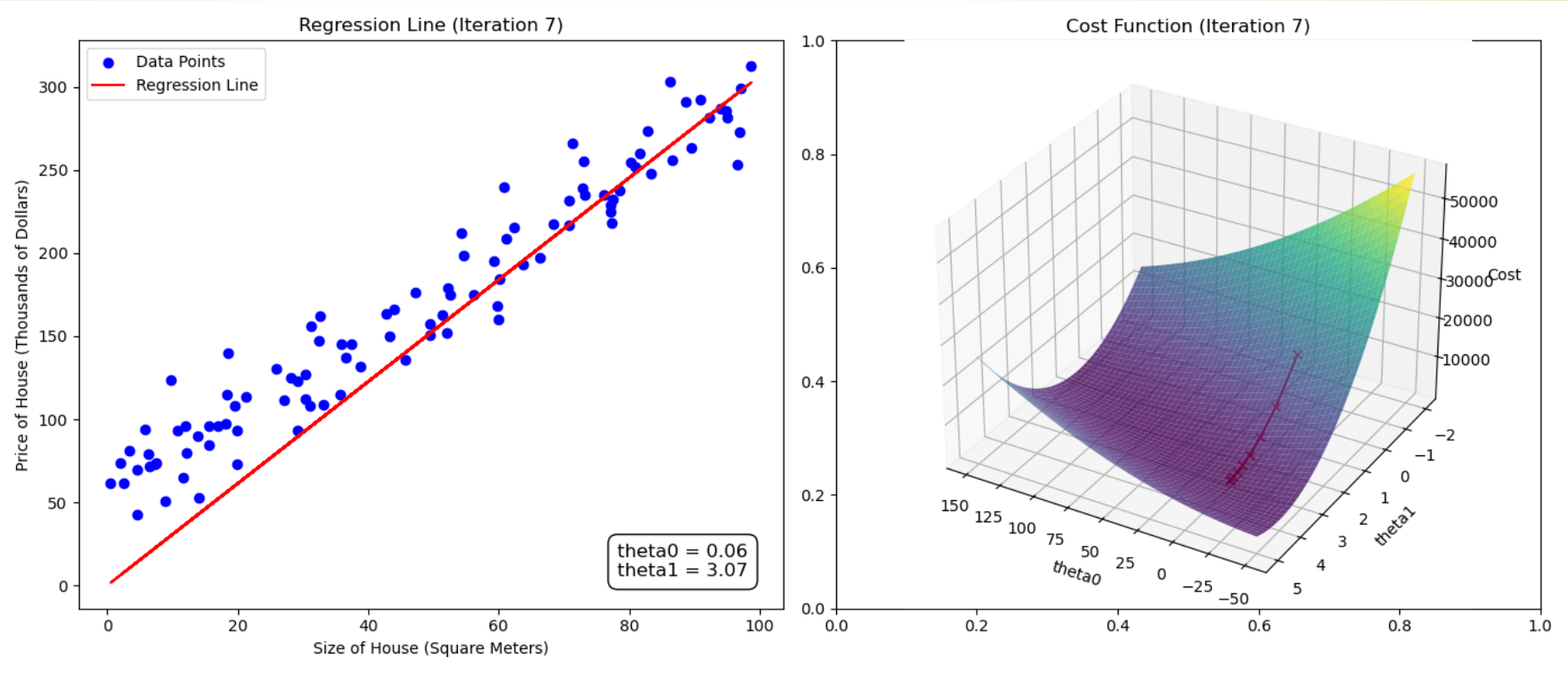

Regression Line (Iteration 2)
Data Points
Regression Line
theta0 = 0.04
theta1 = 2.17
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 2)
Cost
theta0
theta1

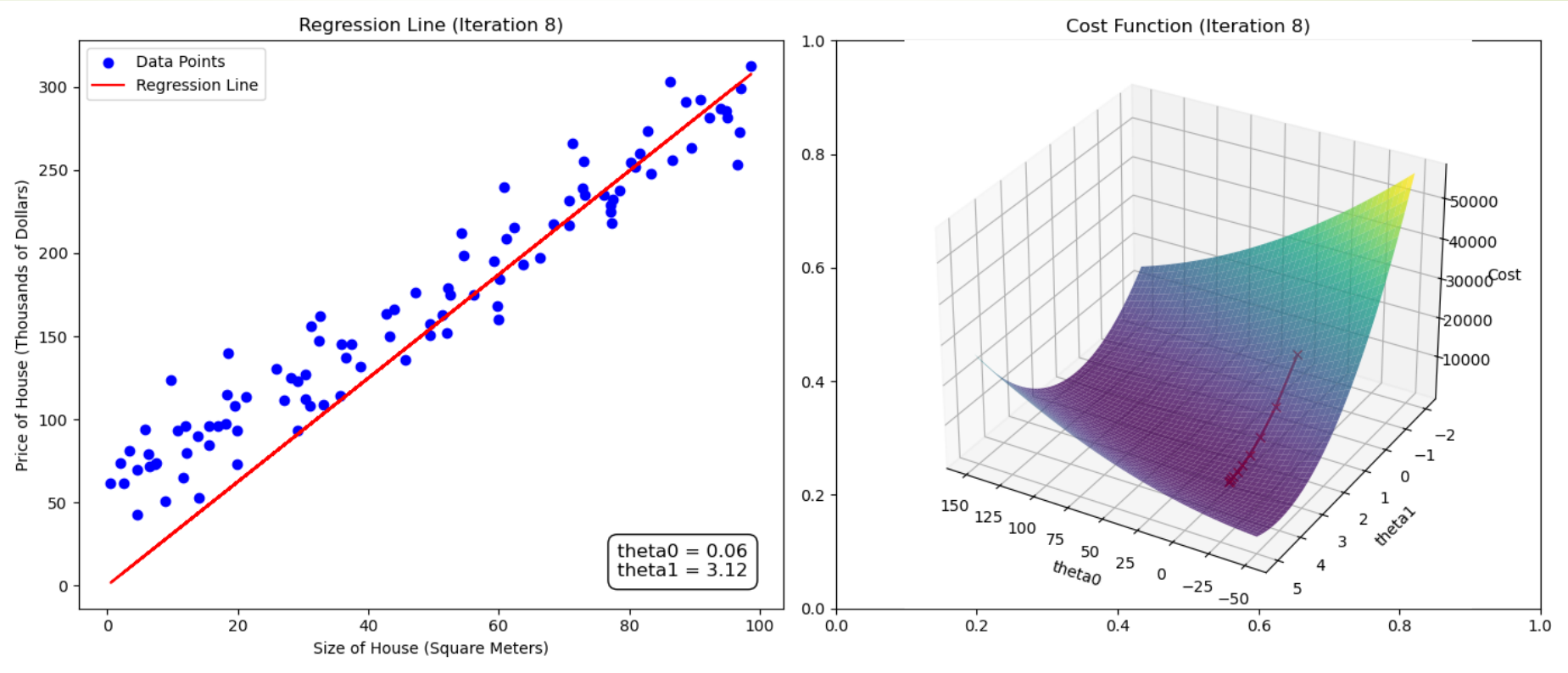

Regression Line (Iteration 3)
Data Points
Regression Line
theta0 = 0.04
theta1 = 2.50
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 3)
Cost
theta0
theta1

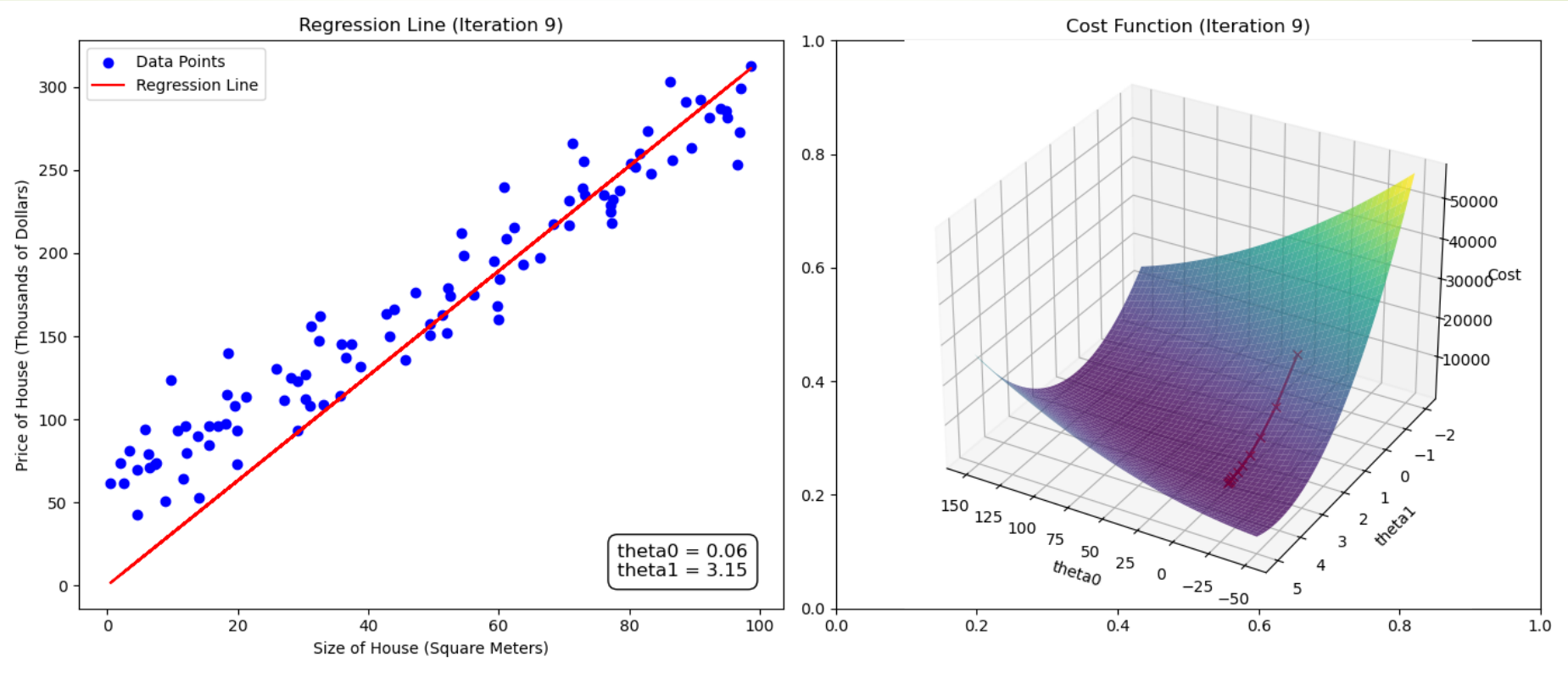

Regression Line (Iteration 4)
Data Points
Regression Line
theta0 = 0.05
theta1 = 2.72
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 4)
Cost
theta0
theta1

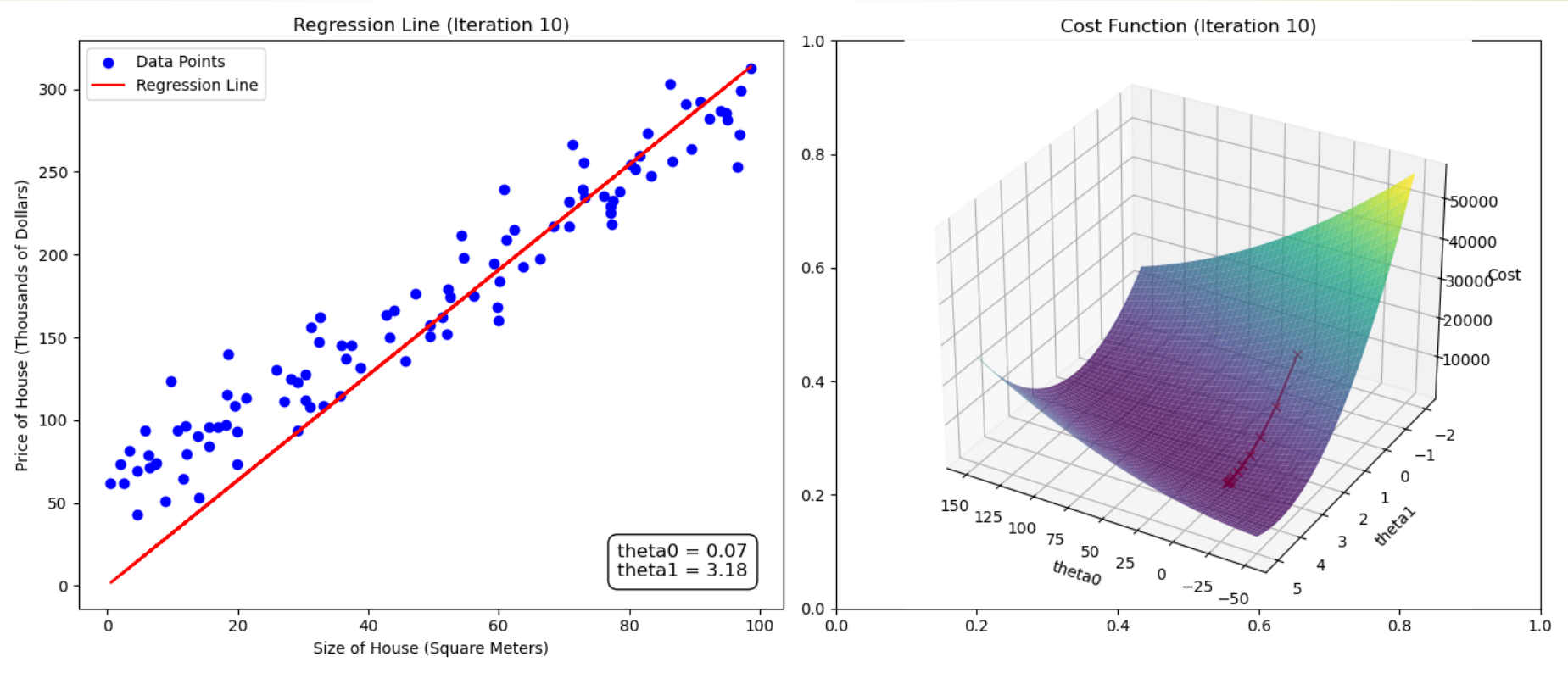

Regression Line (Iteration 5)
Data Points
Regression Line
theta0 = 0.05
theta1 = 2.88
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 5)
Cost
theta0
theta1

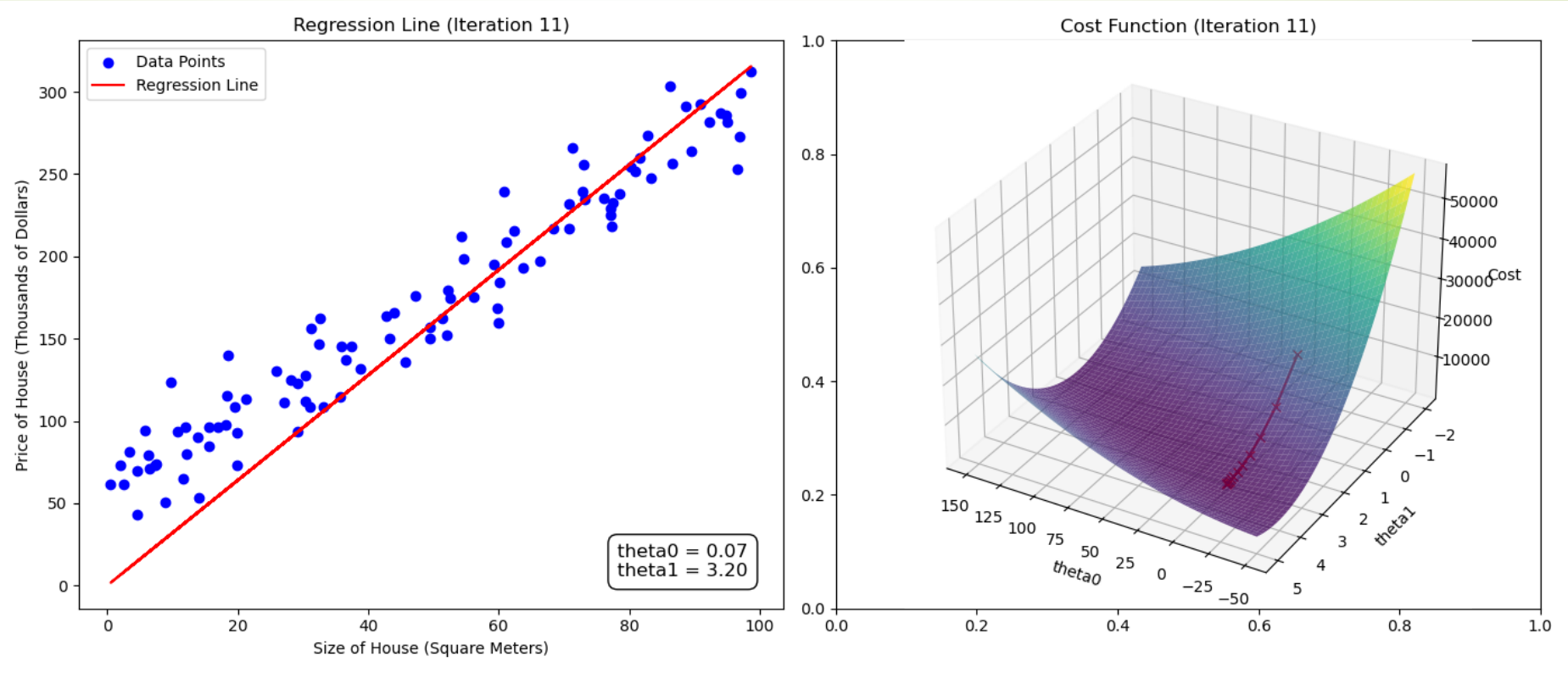

Regression Line (Iteration 6)
Data Points
Regression Line
theta0 = 0.06
theta1 = 2.99
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 6)
Cost
theta0
theta1

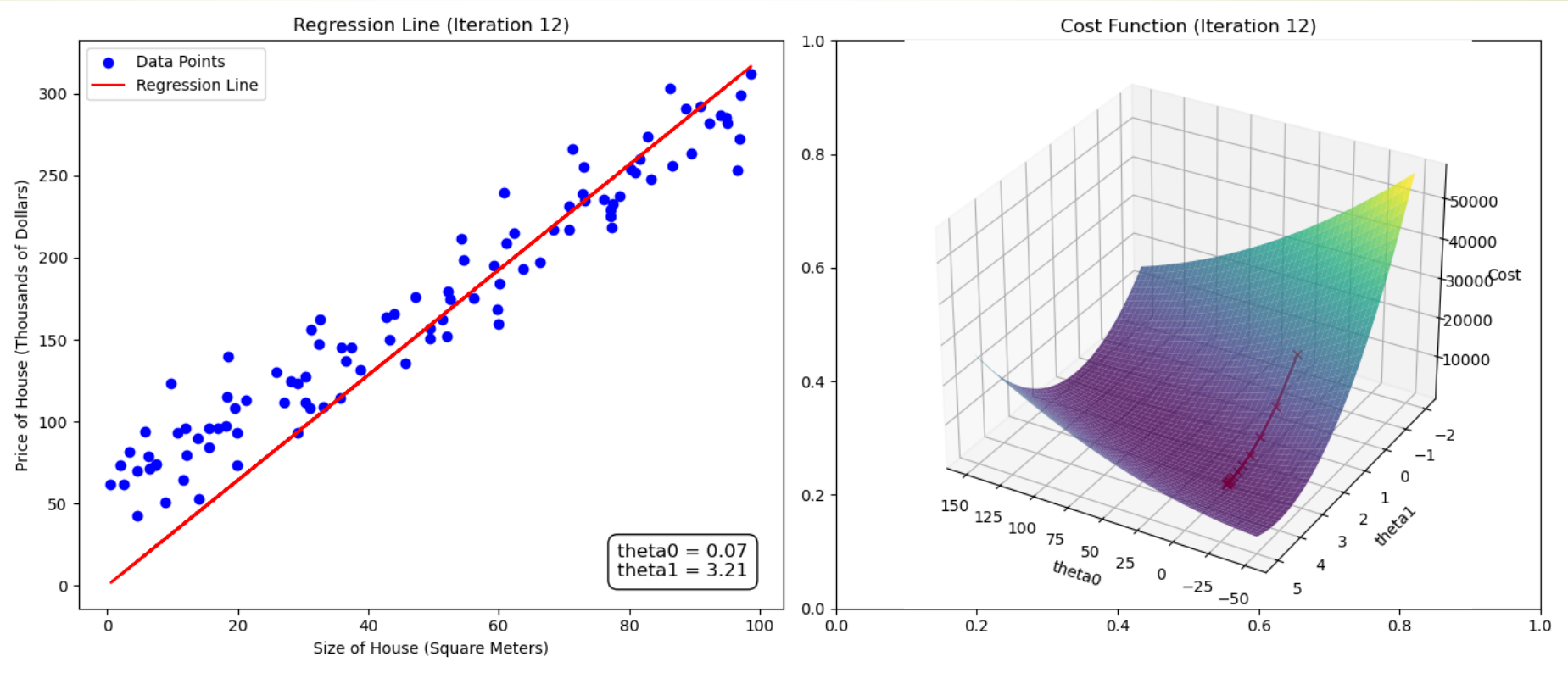

Regression Line (Iteration 7)
Data Points
Regression Line
theta0 = 0.06
theta1 = 3.07
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 7)
Cost
theta0
theta1

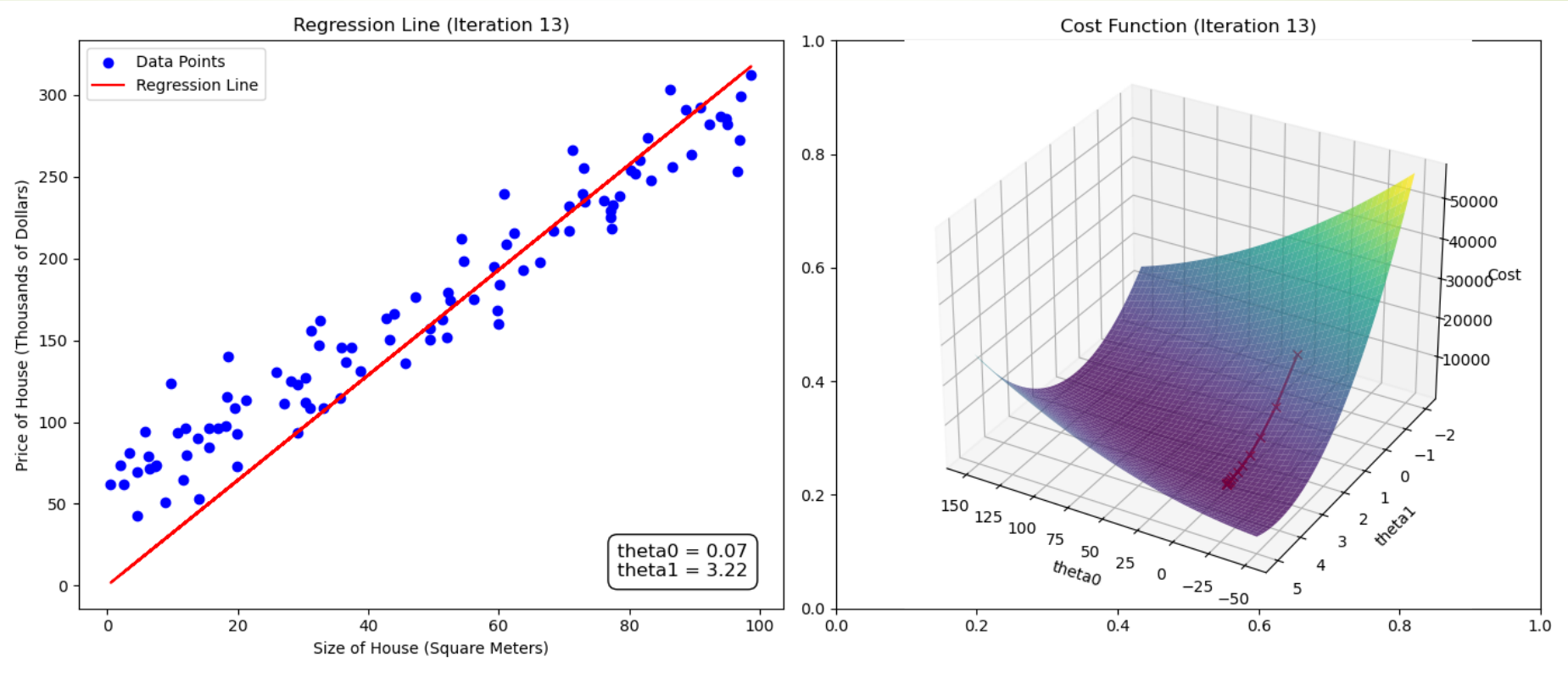

Regression Line (Iteration 8)
Data Points
Regression Line
theta0 = 0.06
theta1 = 3.12
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 8)
Cost
theta0
theta1

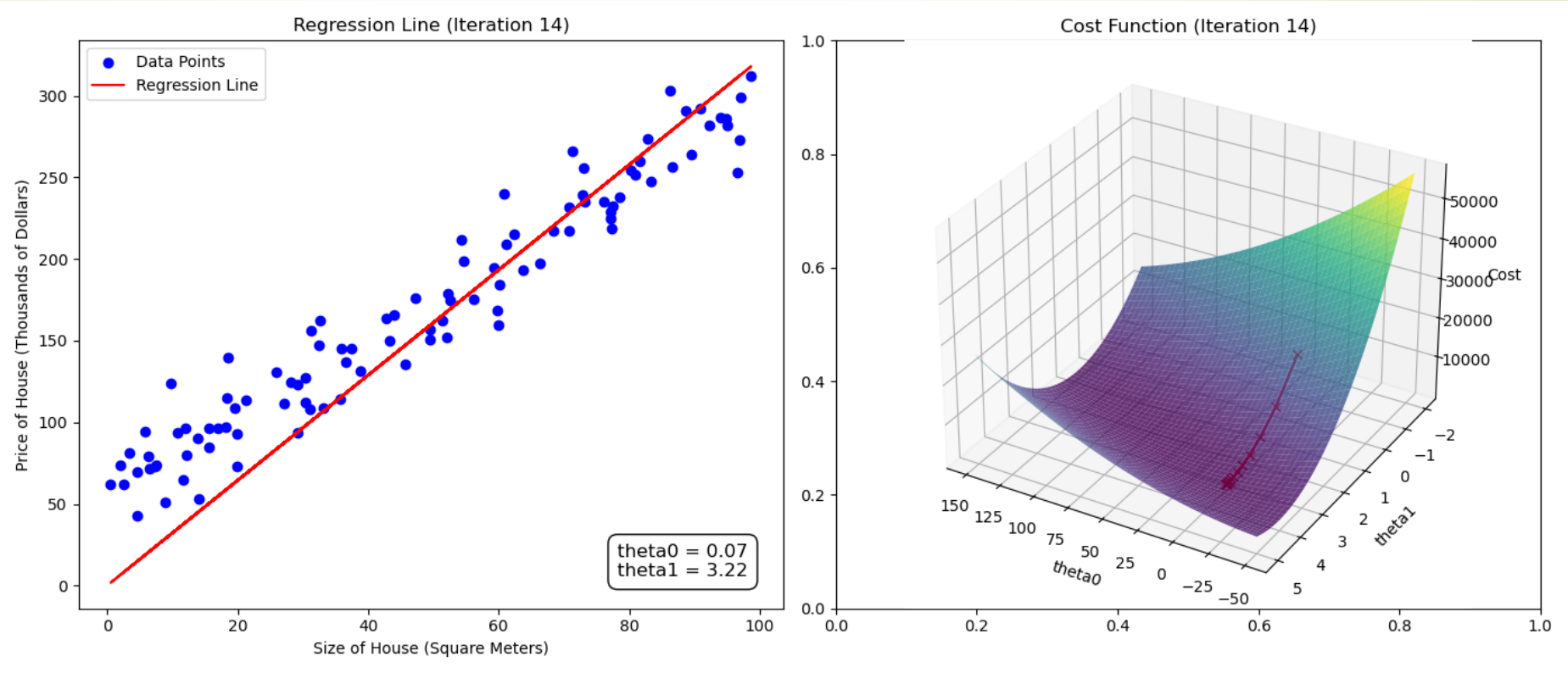

Regression Line (Iteration 9)
Data Points
Regression Line
theta0 = 0.06
theta1 = 3.15
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 9)
Cost
theta0
theta1

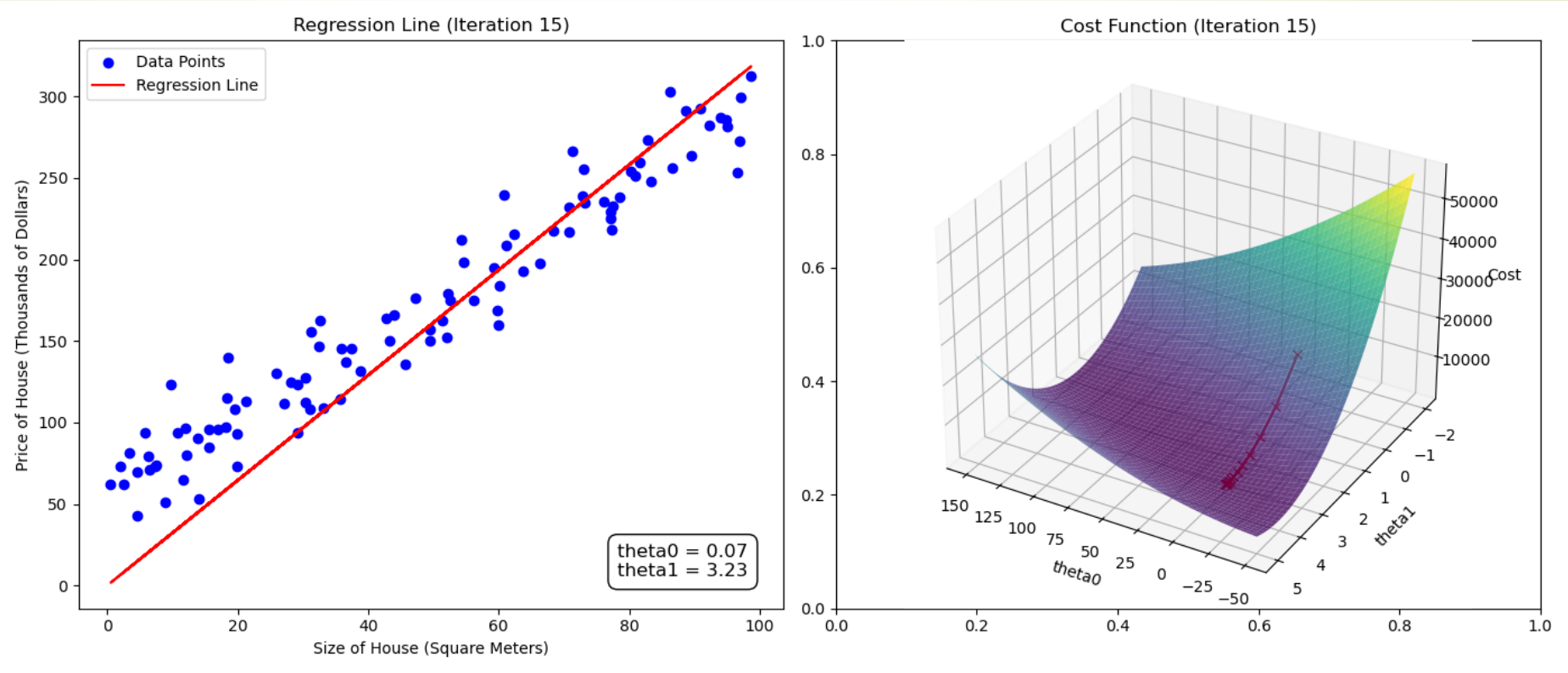

Regression Line (Iteration 10)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.18
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 10)
Cost
theta0
theta1

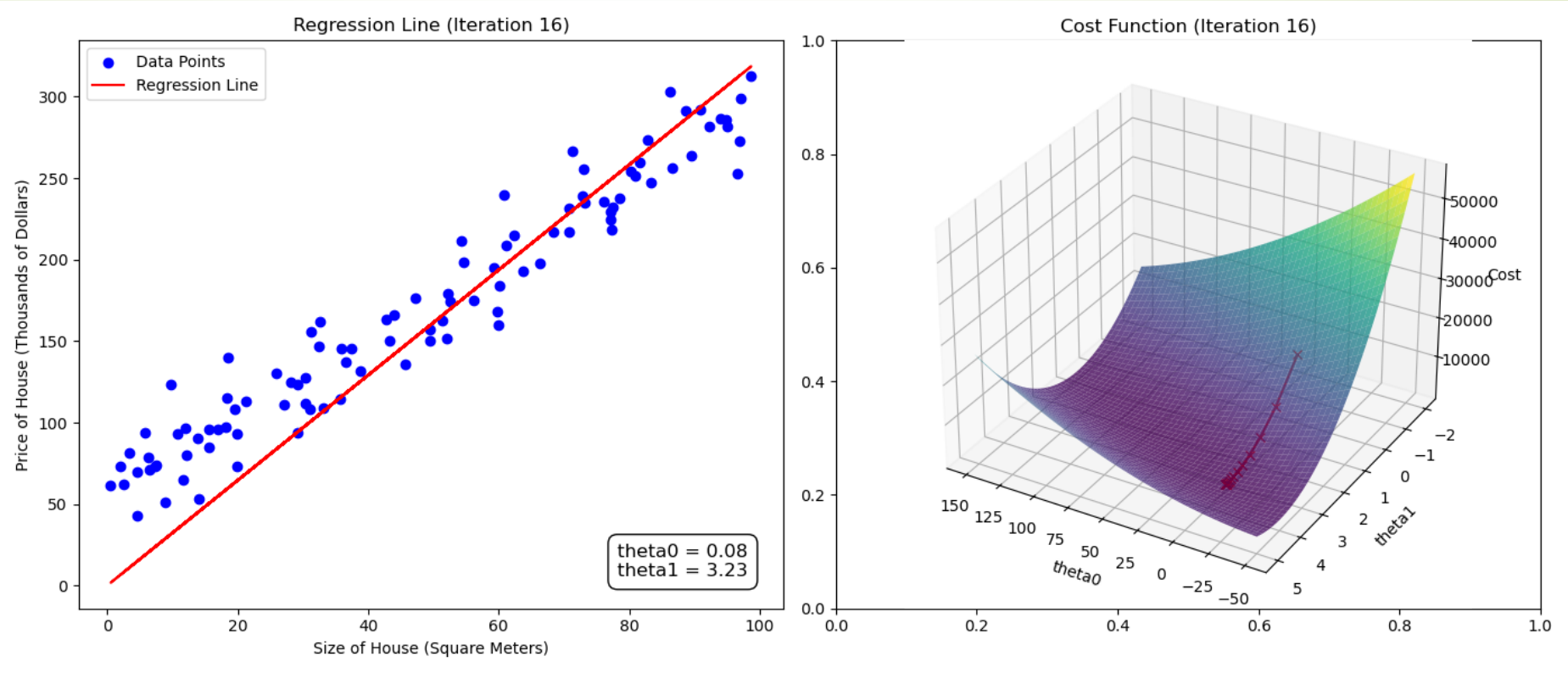

Regression Line (Iteration 11)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.20
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 11)
Cost
theta0
theta1

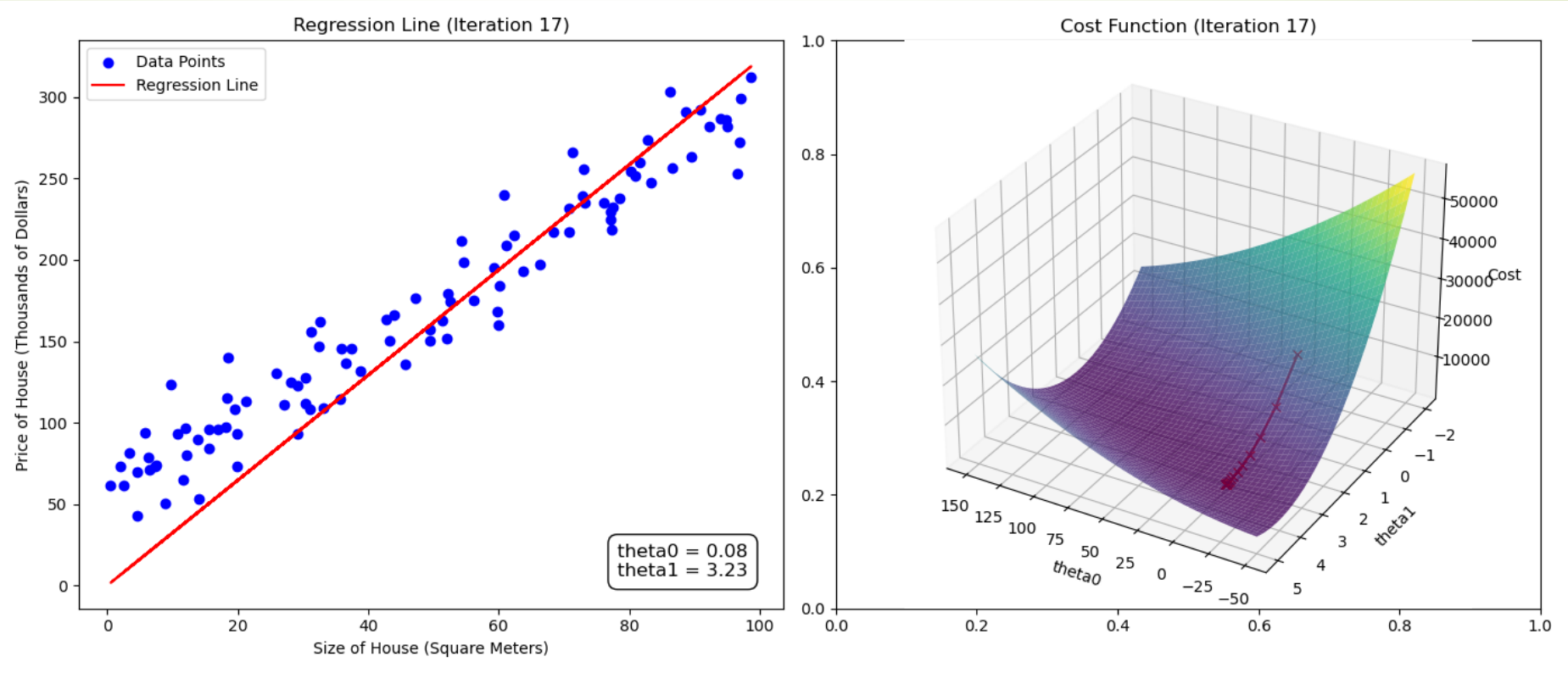

Regression Line (Iteration 12)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.21
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 12)
Cost
theta0
theta1

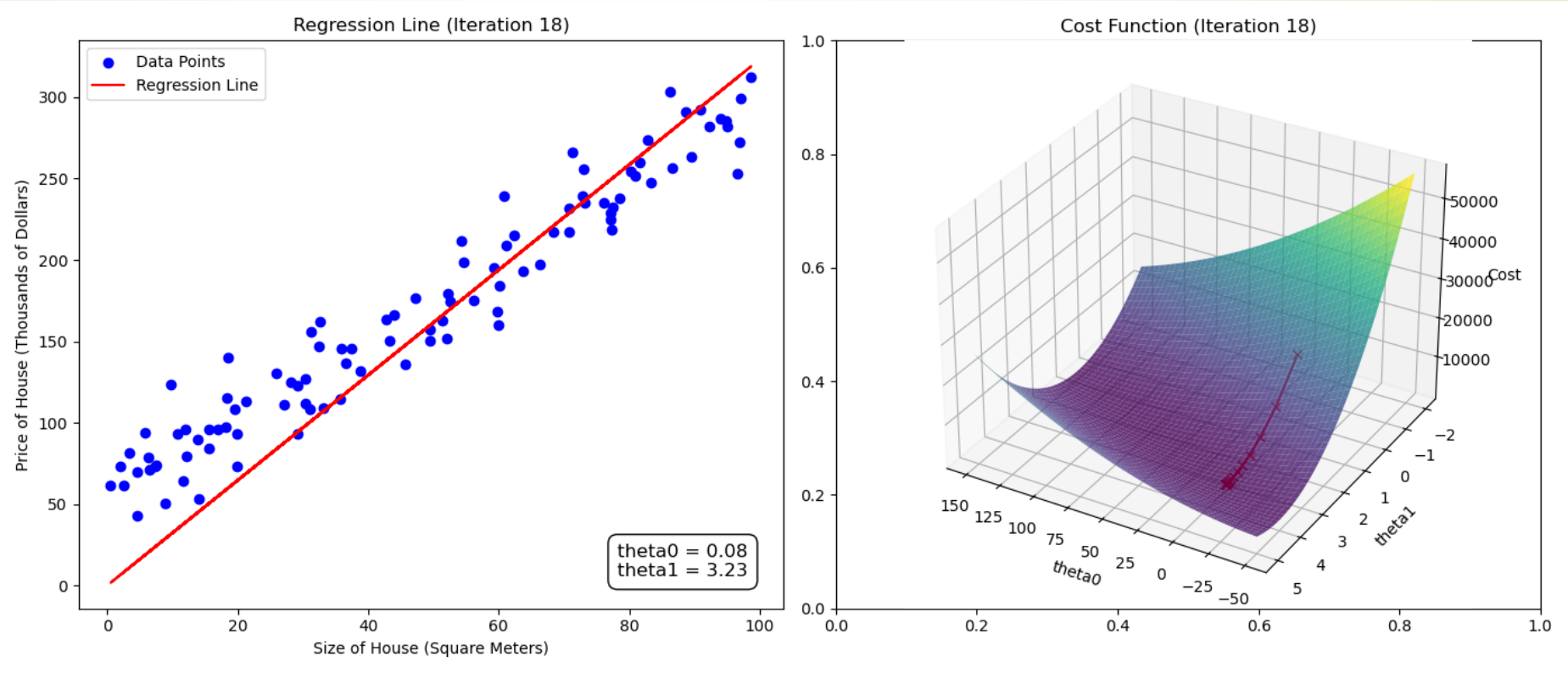

Regression Line (Iteration 13)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.22
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 13)
Cost
theta0
theta1

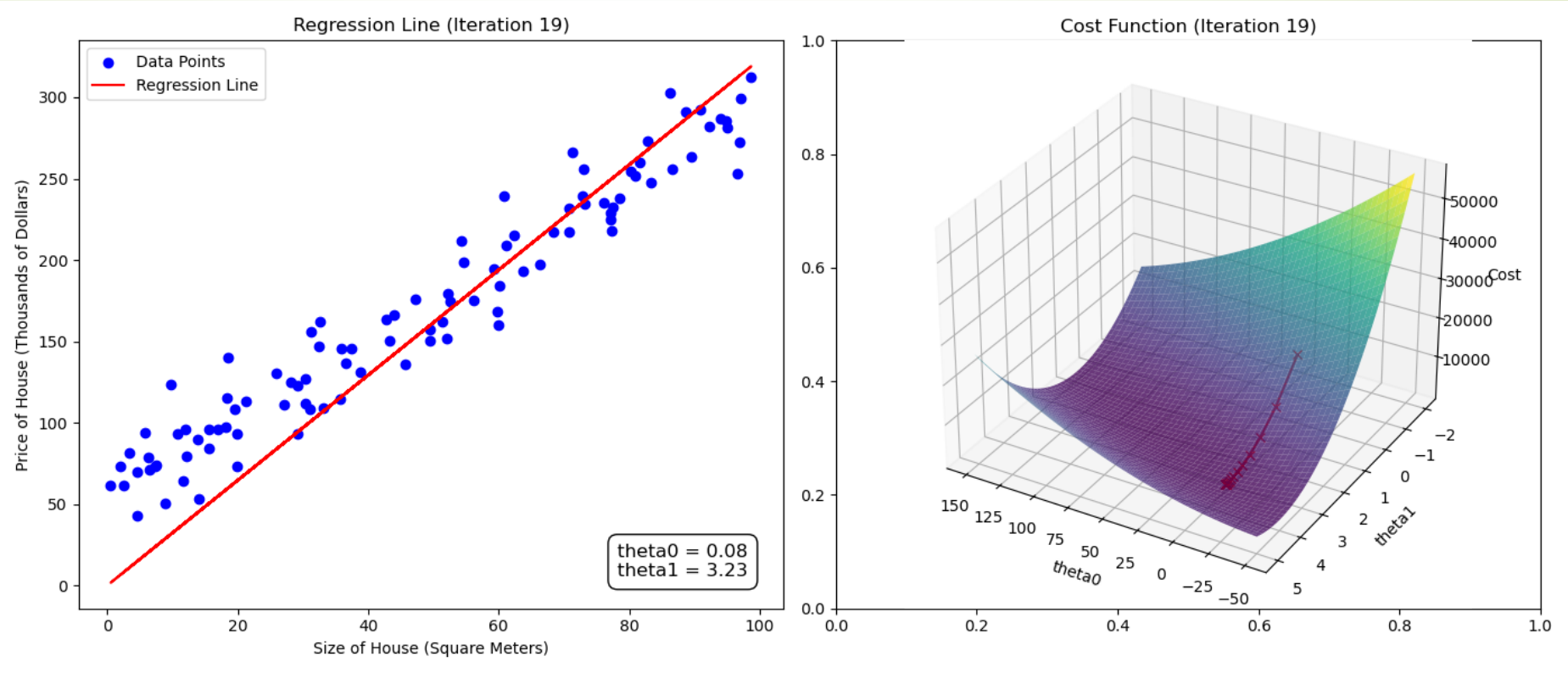

Regression Line (Iteration 14)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.22
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 14)
Cost
theta0
theta1


Regression Line (Iteration 15)
Data Points
Regression Line
theta0 = 0.07
theta1 = 3.23
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 15)
Cost
theta0
theta1

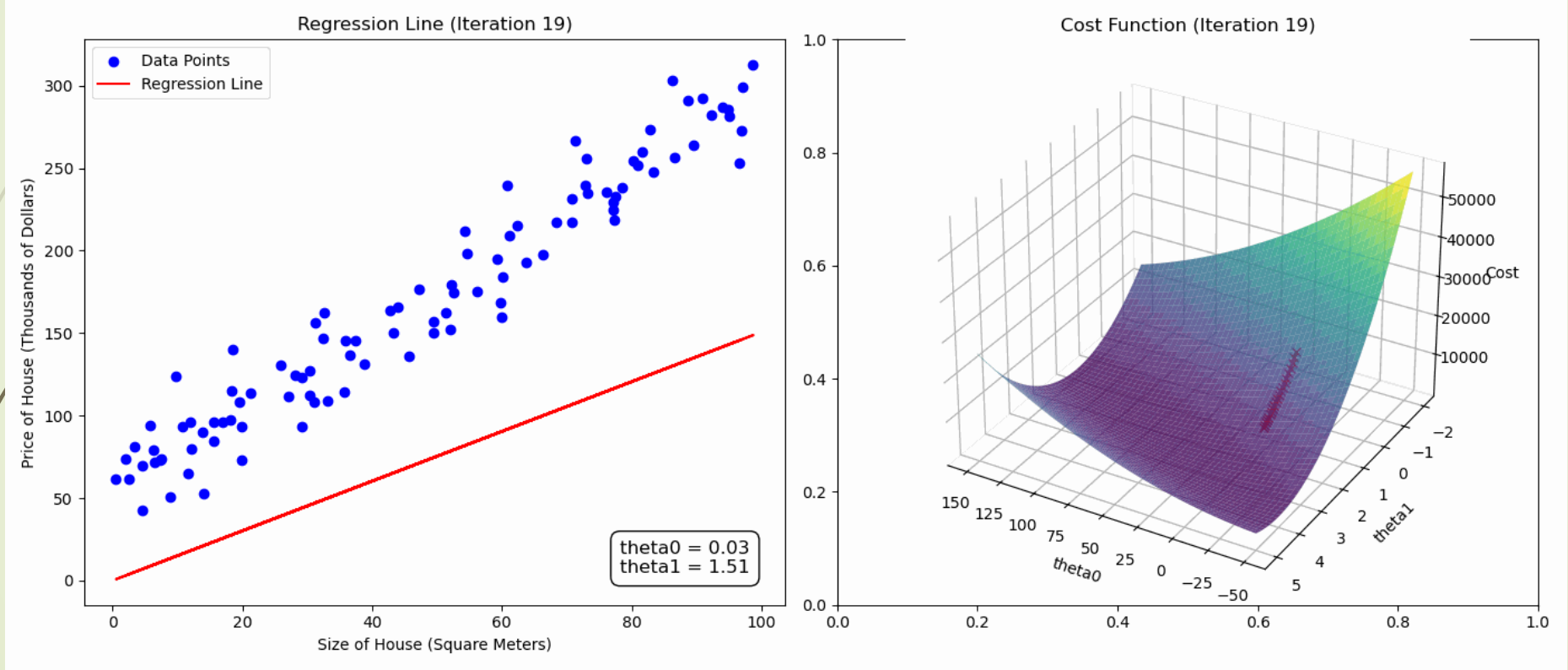

Regression Line (Iteration 16)
Data Points
Regression Line
theta0 = 0.08
theta1 = 3.23
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 16)
Cost
theta0
theta1


Regression Line (Iteration 17)
Data Points
Regression Line
theta0 = 0.08
theta1 = 3.23
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 17)
Cost
theta0
theta1

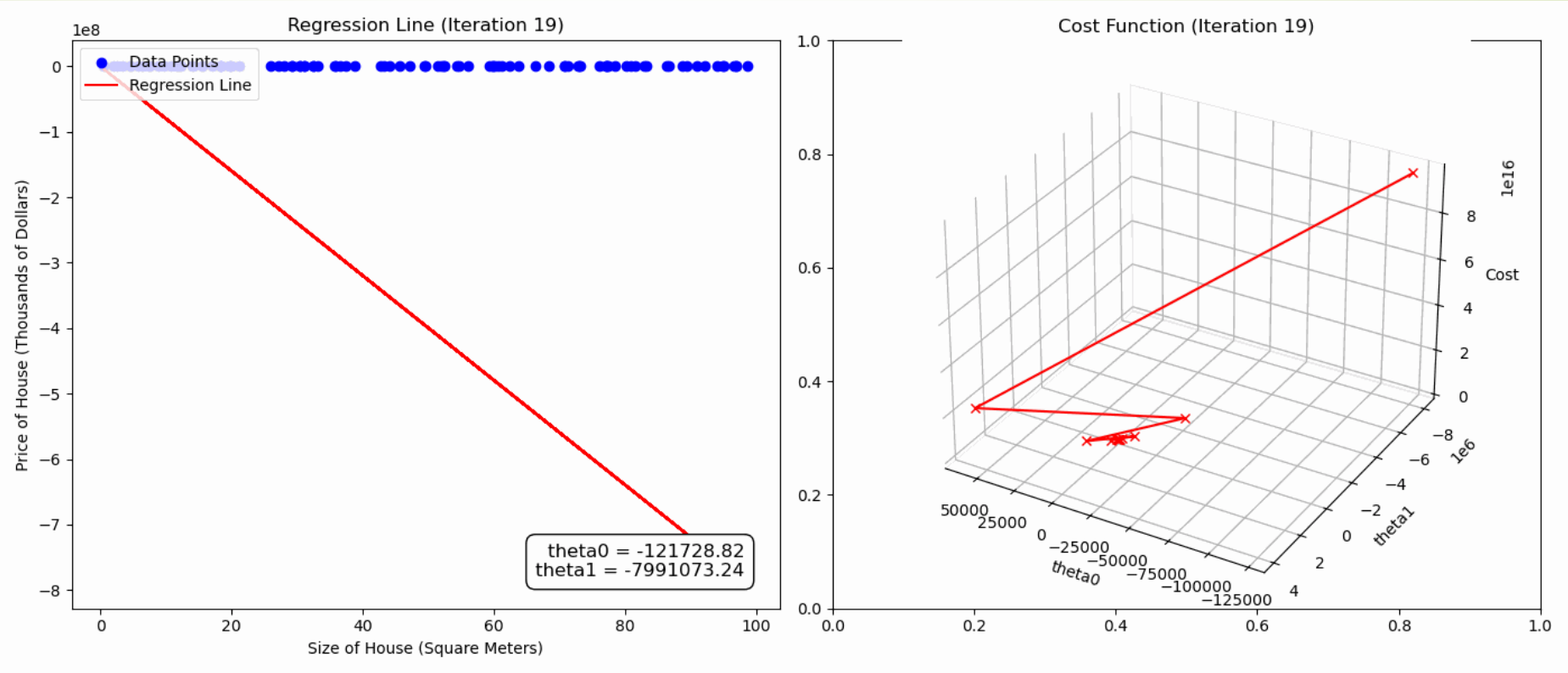

Regression Line (Iteration 18)
Data Points
Regression Line
theta0 = 0.08
theta1 = 3.23
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 18)
Cost
theta0
theta1


Regression Line (Iteration 19)
Data Points
Regression Line
theta0 = 0.08
theta1 = 3.23
Price of House (Thousands of Dollars)
Size of House (Square Meters)
Cost Function (Iteration 19)
Cost
theta0
theta1

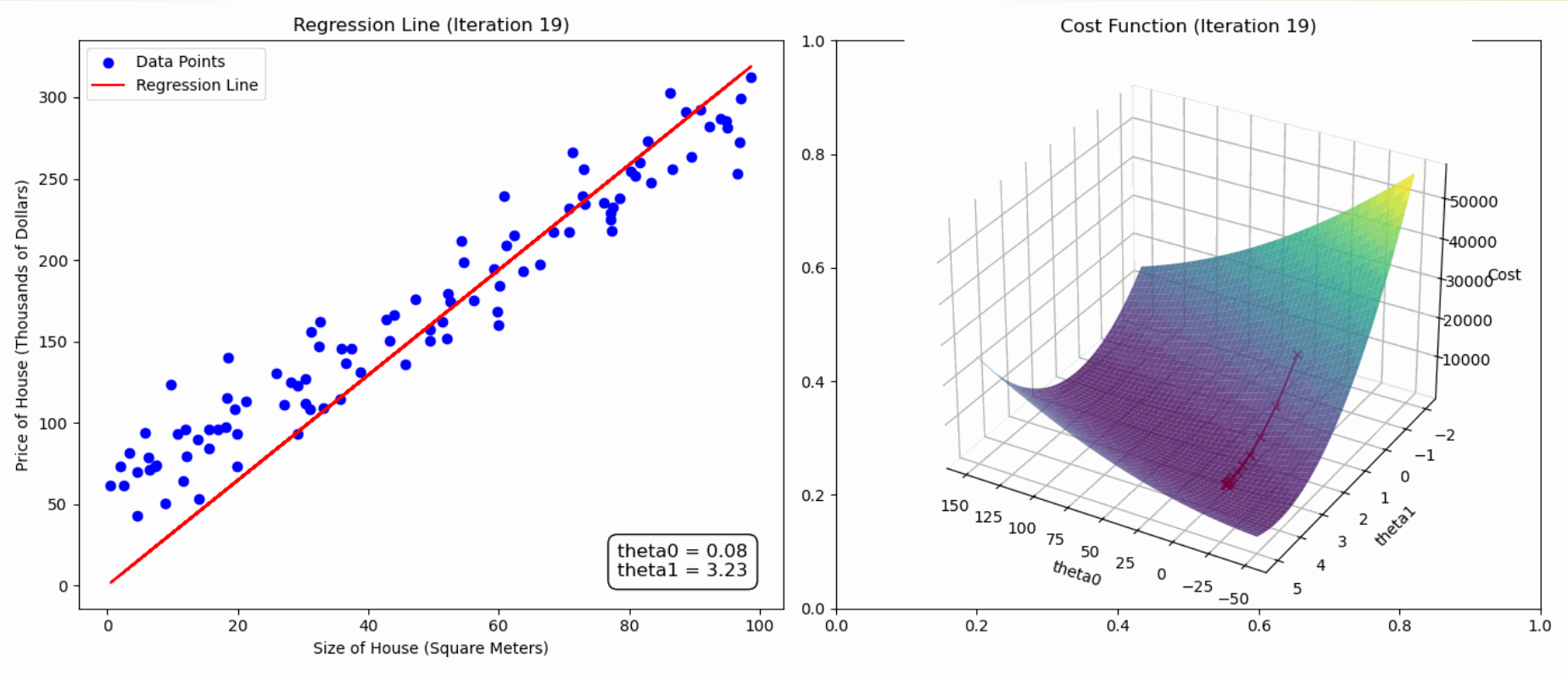
Slower Leaning Rate



Faster Learning Rate – Explode



Proper Learning Rate



Solving Linear Regression with the Normal Equation

- The **Normal Equation** is an analytical method to solve for the parameters θ without using iterative methods like gradient descent.

- The equation for the **Normal Equation** is:

$$\theta = (X^T X)^{-1} X^T y$$

- Where:
  - X is the matrix of feature variables.
  - y is the vector of target values (dependent variable).
  - θ is the vector of parameters we are solving for.

Why Use the Normal Equation?

- **No need for learning rate:** Unlike gradient descent, the normal equation doesn't require tuning a learning rate.
- **Exact Solution:** Provides a closed-form solution without requiring iterations.
- **Steps to Solve:**

1. **Create the Design Matrix** X: Add a column of ones to represent θ_0 (intercept term).
2. **Compute** $X^T X$: Multiply the transpose of the design matrix X by itself.
3. **Compute** $(X^T X)^{-1}$: Find the inverse of the product.
4. **Compute** $X^T y$: Multiply the transpose of X by the target vector y.
5. **Multiply**: Use the equation $\theta = (X^T X)^{-1} X^T y$ to get the parameters θ.

Normal Equation vs Gradient Descent

Normal Equation

- No need to choose α (Learning rate)
- It is an analytical method and provides a solution in one shot.
- You need to compute $(X^TX)^{-1}$ an nxn matrix, where n is the number of features.
- **Computational Complexity**: $O(n^3)$ meaning it becomes computationally expensive for large n.
- Only works for least squares loss.

Gradient Descent:

- You need to **choose** α (Learning rate)
- It is an **iterative** approach that updates parameters until convergence.
- Works for **almost every loss function** (not limited to least squares).
- Efficient for datasets with **large numbers of features n**, as it does not require matrix inversion.

Non-Invertible $(X^TX)^{-1}$ in the Normal Equation

- The normal equation relies on the term $(X^TX)^{-1}$, the **inverse** of (X^TX)
- **Problem**: If (X^TX) is **not invertible** (i.e., we cannot calculate its inverse), this means that there is an issue with the features in the dataset.
- Why Does This Happen?
  - **Linearly Dependent Features**:
    - Definition: If one or more features are a linear combination of other features, then (X^TX) becomes singular (non-invertible).
  - **Too Many Features:**
    - If the number of features is **greater than** the number of observations, then (X^TX) will also be non-invertible.
    - This situation is referred to as overfitting, where there is too much complexity in the model.

How did we get the Normal Equation

- Cost Function in Matrix form:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}(X\theta - y)^\top (X\theta - y)$$

- Expanding the cost function:

$$J(\theta) = \frac{1}{2m}\left(\theta^\top X^\top X\theta - \theta^\top X^\top y - y^\top X\theta + y^\top y\right)$$

- Since $\theta^T X^T y$ is a scalar, it equals its transpose, so:

$$J(\theta) = \frac{1}{2m}\left(\theta^\top X^\top X\theta - 2y^\top X\theta + y^\top y\right)$$

- To minimize J(θ), take the gradient of J with respect to θ and set it to zero

$$\nabla_\theta J(\theta) = \frac{1}{2m}\left(2X^\top X\theta - 2X^\top y\right) = 0$$

- Simplifying, we find the normal equation:

$$X^\top X\theta = X^\top y$$

- The solution to this equation is:

$$\theta = (X^\top X)^{-1} X^\top y$$

Assumptions

Assumptions in Multiple Linear Regression

- **What is Multicollinearity?**
  - **Definition**: Multicollinearity occurs when two or more independent variables in a regression model are highly correlated, meaning they provide overlapping information.
  - **Why it matters:** Ideally, each predictor (independent variable) should contribute unique information to explain the dependent variable. When variables are highly correlated, it becomes difficult to isolate the individual effect of each predictor on the outcome.

Assumptions in Multiple Linear Regression

- **Additivity**: The effect of each predictor variable on the dependent variable is additive and does not depend on the values of other variables.
- **Key Point**: Each variable independently contributes to the response without interacting with other variables.
- **Why It Matters:**
  - **No Interaction Effects:** In standard multiple linear regression, the model assumes no interaction between predictors.

Assumptions in Multiple Linear Regression

- **Feature Selection:** Choosing the right features is critical for an effective multiple linear regression model.
- **Irrelevant Features:** Including features that do not contribute to the prediction can lead to overfitting and make the model less interpretable.
- **Redundant Features:** Redundant variables (i.e., highly correlated predictors) should be removed to avoid multicollinearity.
- **Feature Selection Techniques:**
  - **Forward Selection:** Start with no variables and add them one by one based on improvement in the model.
  - **Backward Elimination:** Start with all variables and remove the least significant ones step by step.
  - **Lasso Regression**: Regularization method that automatically selects and removes irrelevant features by penalizing large coefficients.

Overfitting

- **Overfitting:** Overfitting happens when the model fits the training data too closely, capturing noise or random fluctuations rather than the underlying pattern.
- **Why It Matters:** A model that is too complex performs well on training data but generalizes poorly to new, unseen data.
- **How to Prevent Overfitting:**
  - **Regularization:** Add a penalty to prevent overfitting by shrinking large coefficients.
  - **Cross-Validation:** Use cross-validation to test the model's performance on unseen data and adjust complexity accordingly.
  - **Simplification:** Avoid including too many variables or overly complex interactions that don't significantly improve the model.