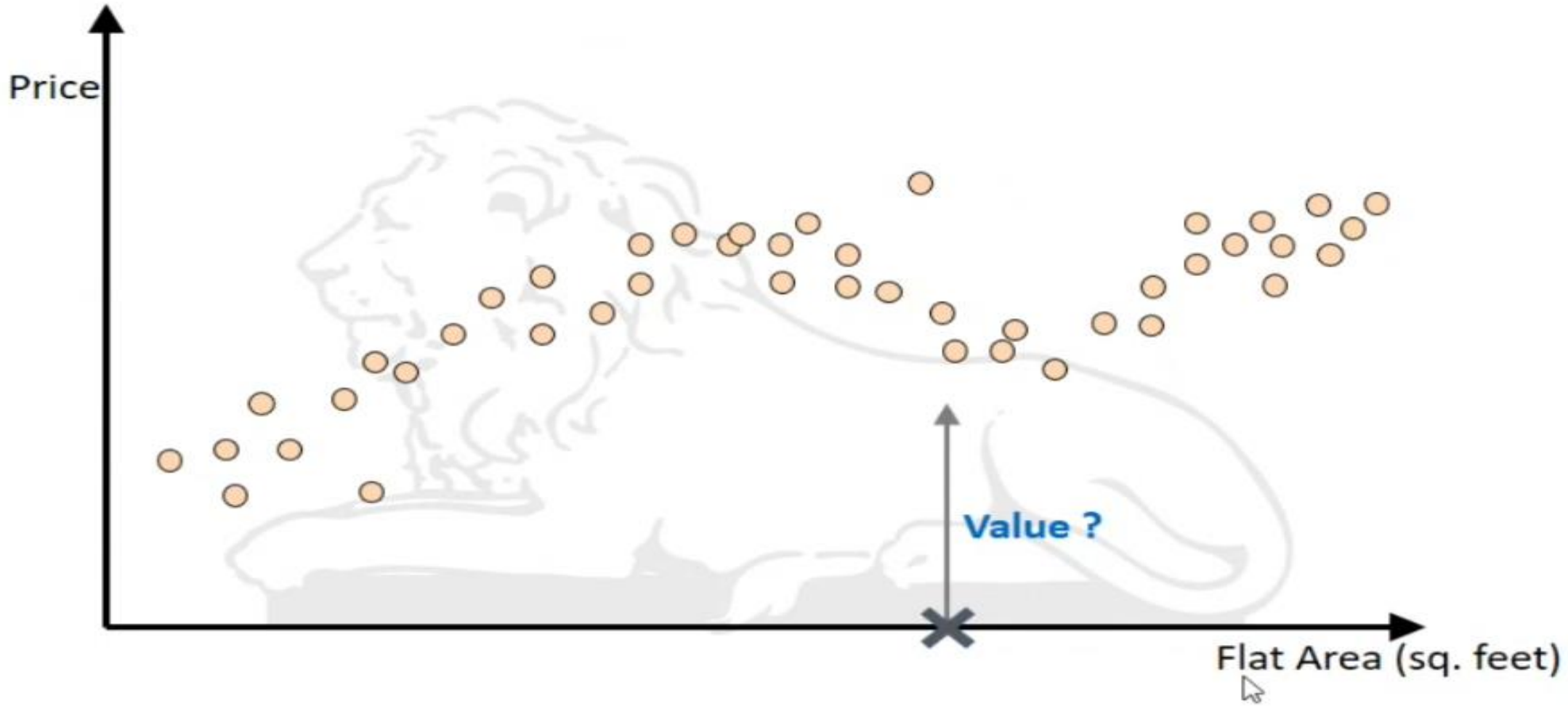
Linear Regression

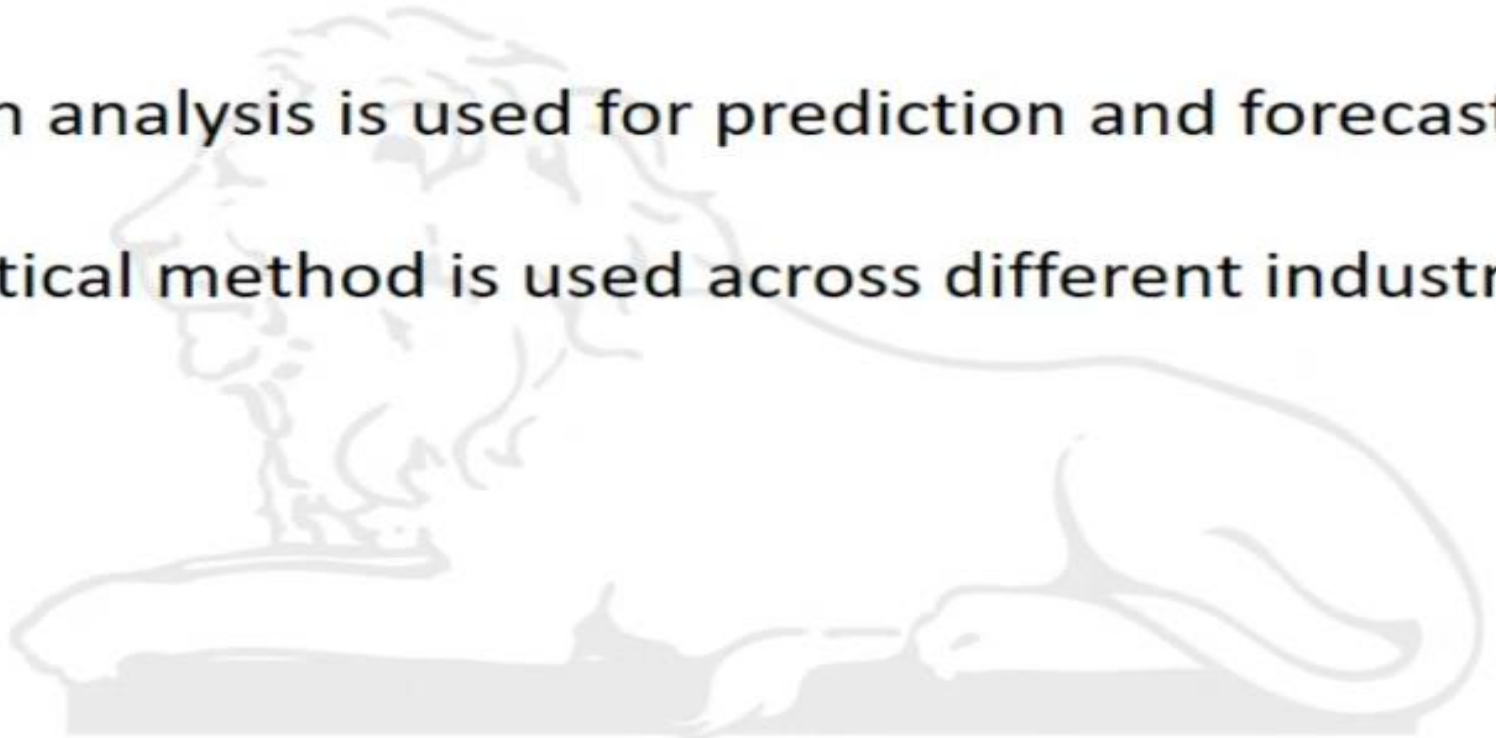
Regression



What is Regression?

Regression analysis is a statistical method that helps us to analyze and understand the relationship between two or more variables of interest.

- Regression analysis is used for prediction and forecasting.
- This statistical method is used across different industries such as:



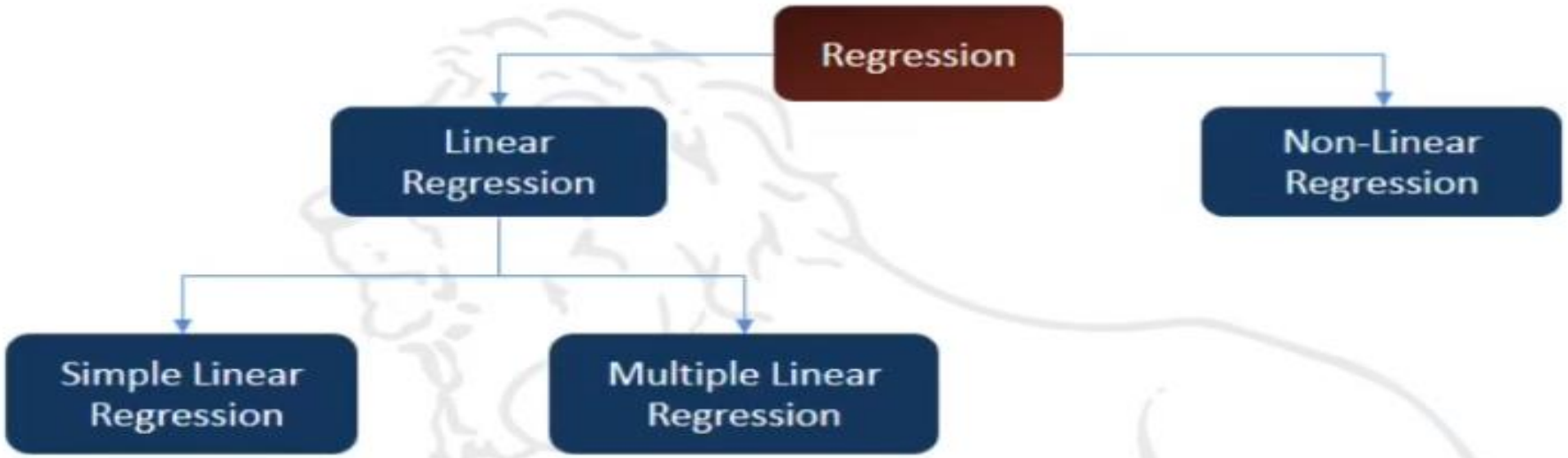
What is Regression?

Regression analysis is a statistical method that helps us to analyze and understand the relationship between two or more variables of interest.

- Regression analysis is used for prediction and forecasting.
- This statistical method is used across different industries such as:

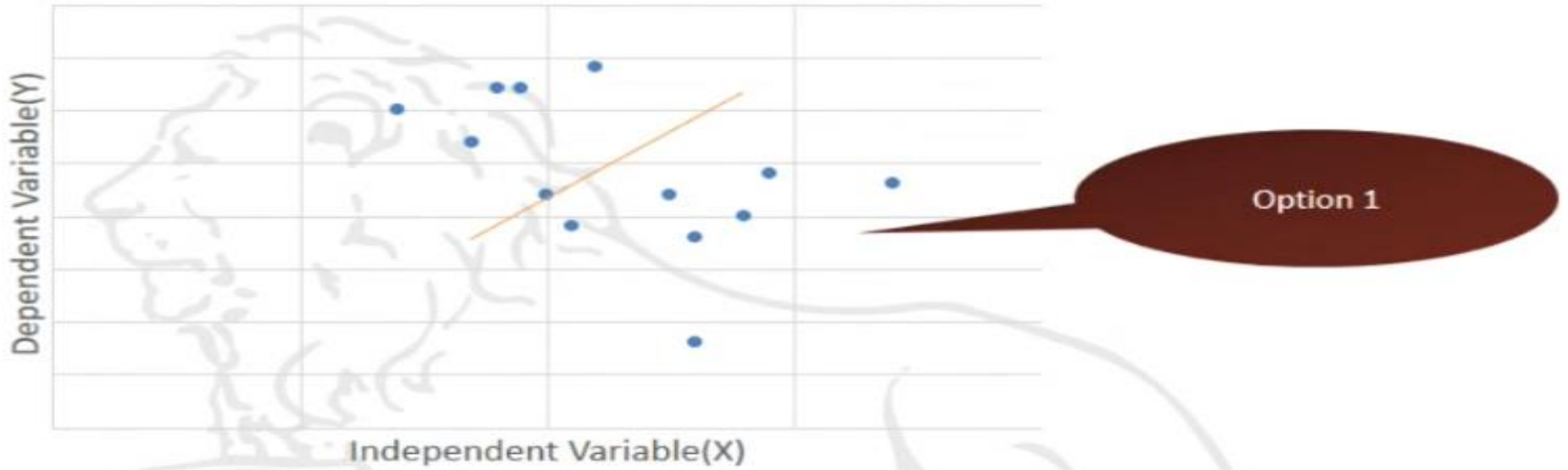
  - Financial Industry
  - Manufacturing
  - Marketing
  - Medicine

Types of Regression



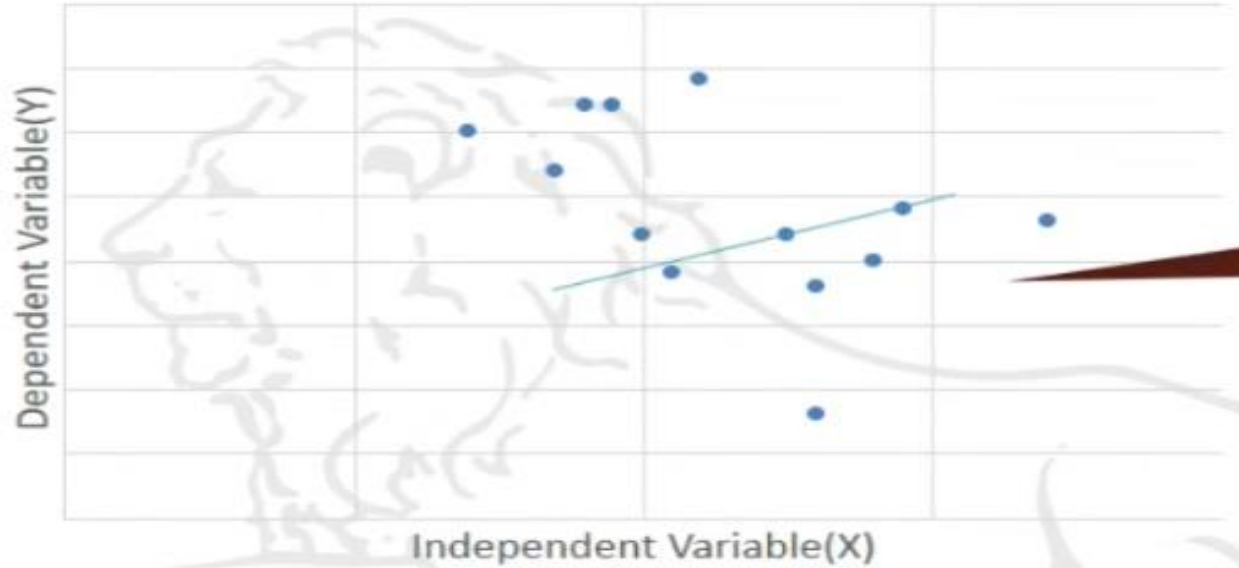
What is Linear Regression?

Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



What is Linear Regression?

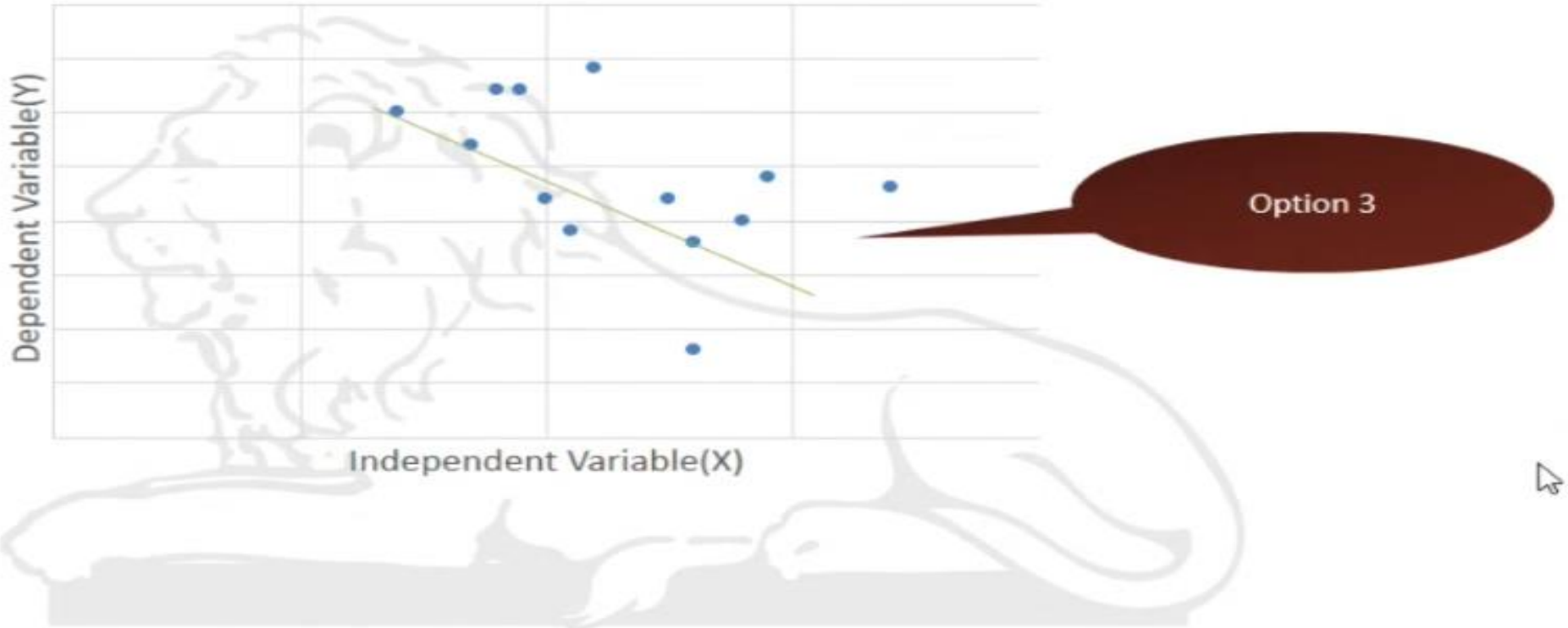
Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



Option 2

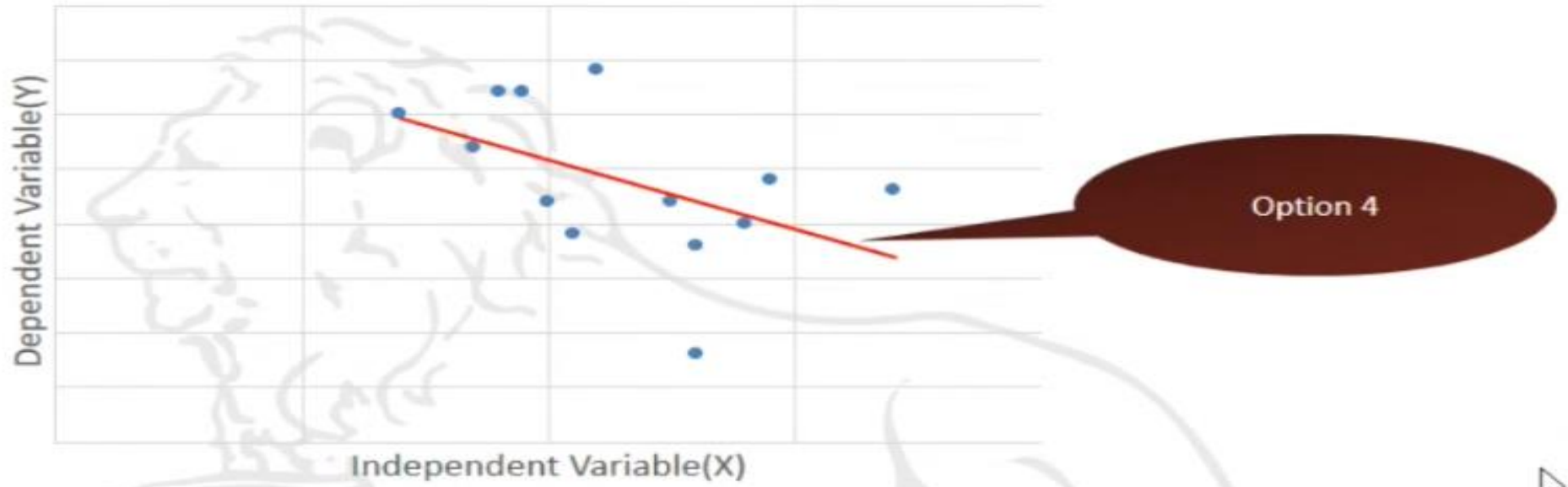
What is Linear Regression?

Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



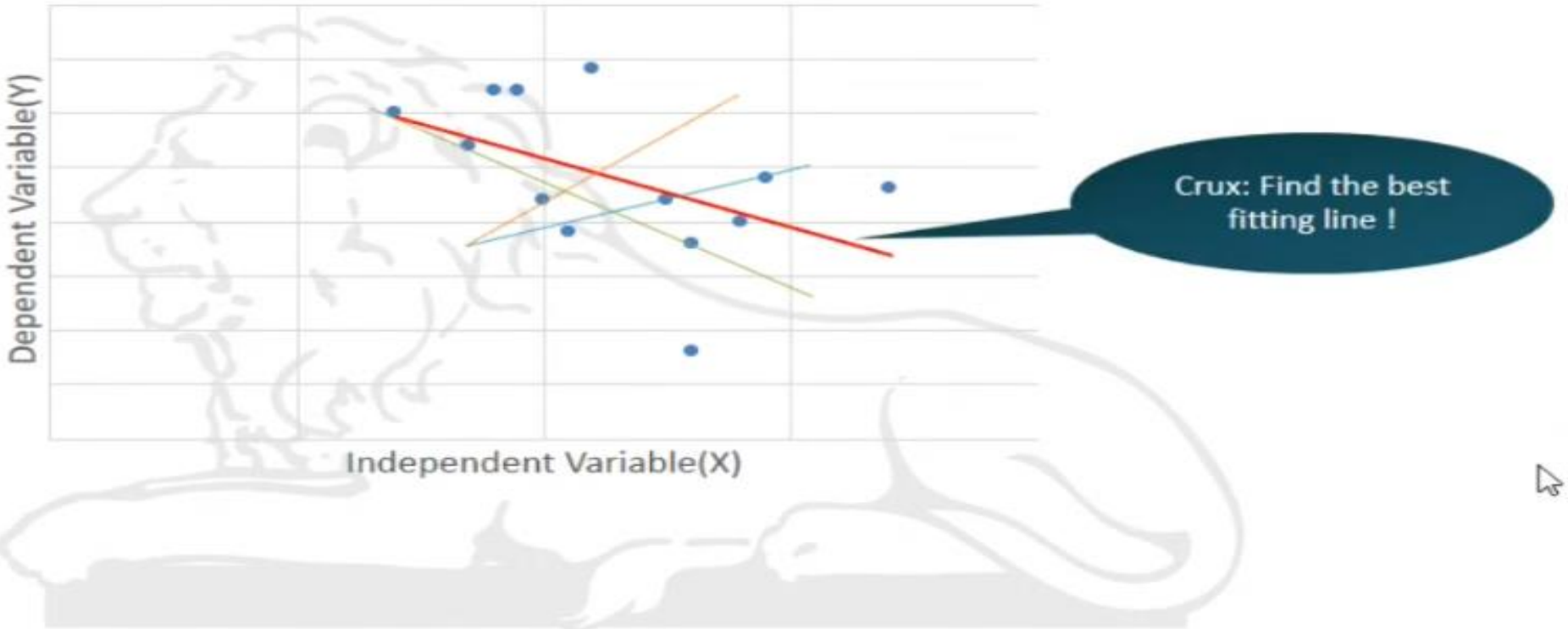
What is Linear Regression?

Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



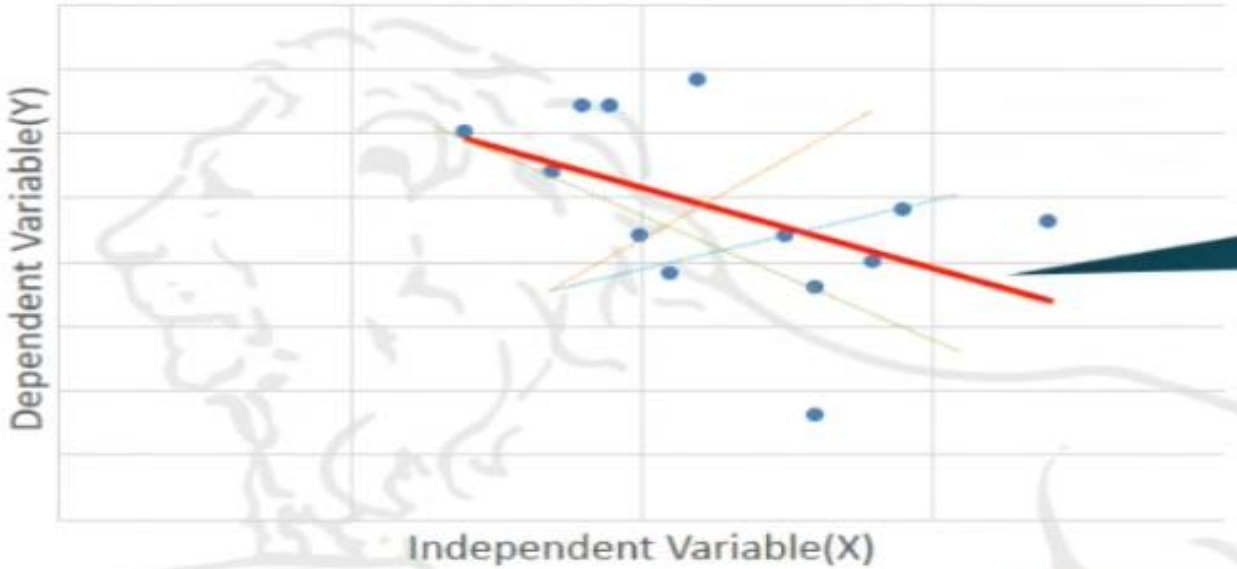
What is Linear Regression?

Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



What is Linear Regression?

Linear Regression analysis is a form of predictive modelling technique which investigates the *linear* relationship between a **dependent** and **independent variable**.



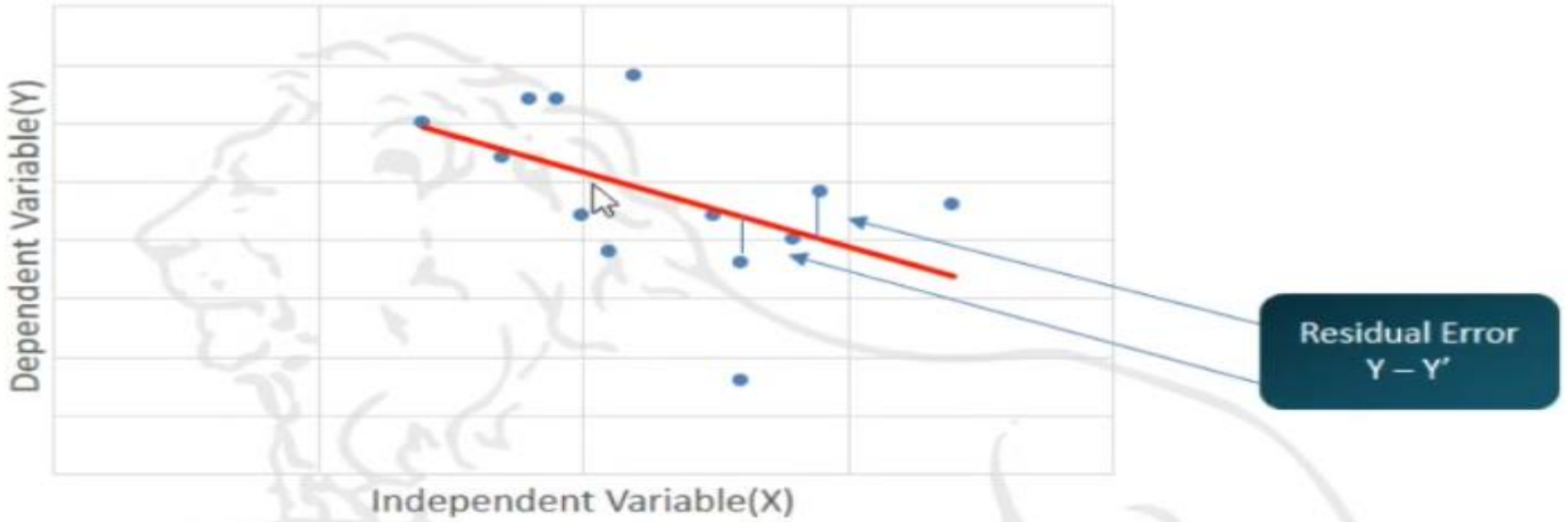
Crux: Find the best fitting line, i.e. Find the best coefficients!

Obviously, even the best fitting line is not perfect!

The Best fitting line is the one with minimal RESIDUAL ERROR .

Residual Error

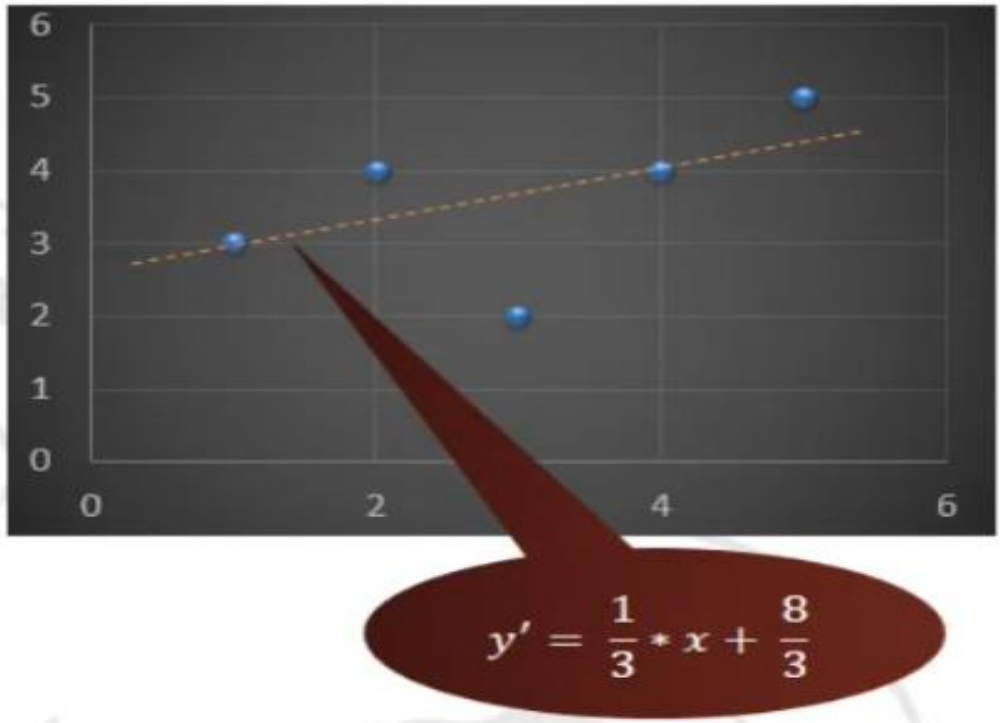
Residuals are the deviations of predicted values from observed values.



Residual Error - Example

Consider an example below :

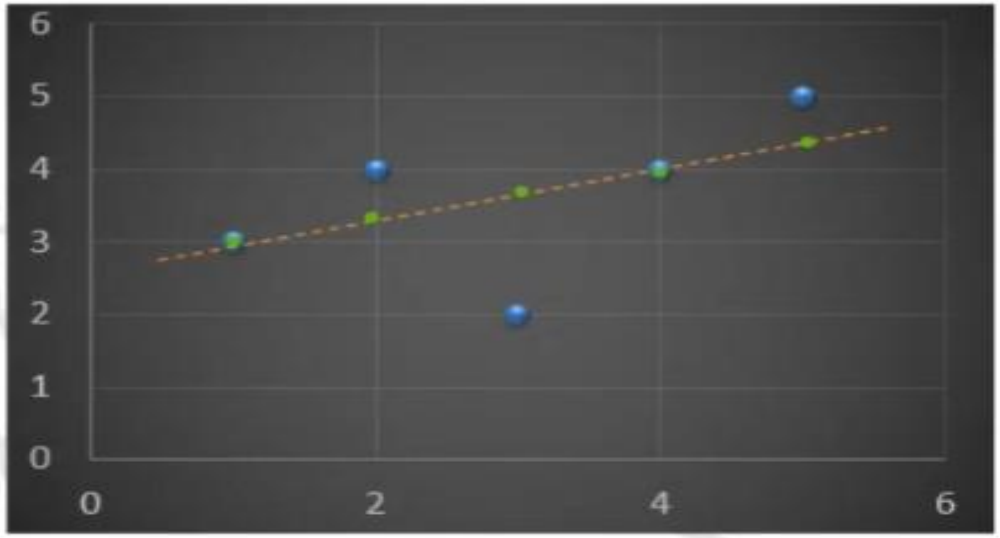
X	Y
1	3
2	4
3	2
4	4
5	5



Residual Error - Example

Consider an example below :

X	Y	Y'
1	3	3
2	4	3.2
3	2	3.6
4	4	4
5	5	4.3



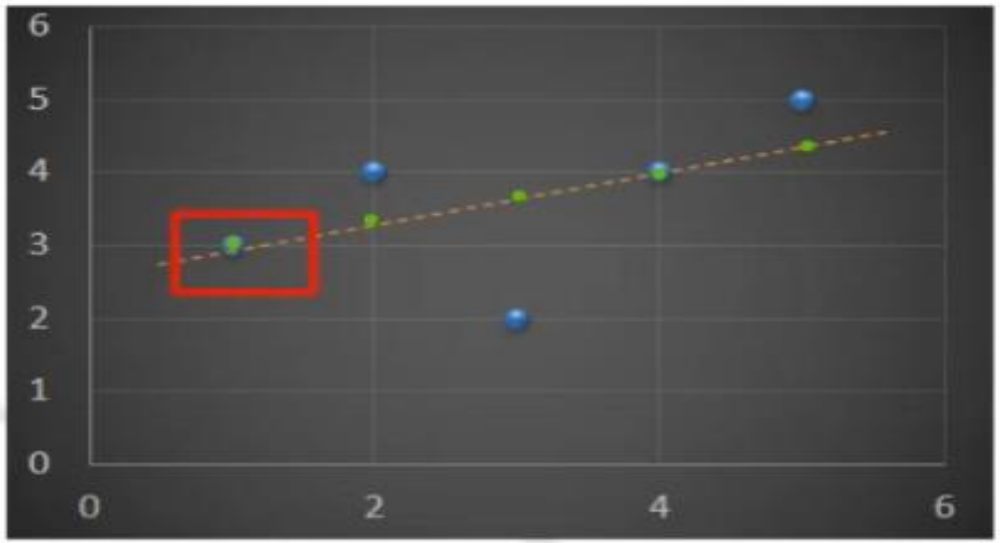
Expected Values

Predicted Values

Residual Error - Example

Consider an example below :

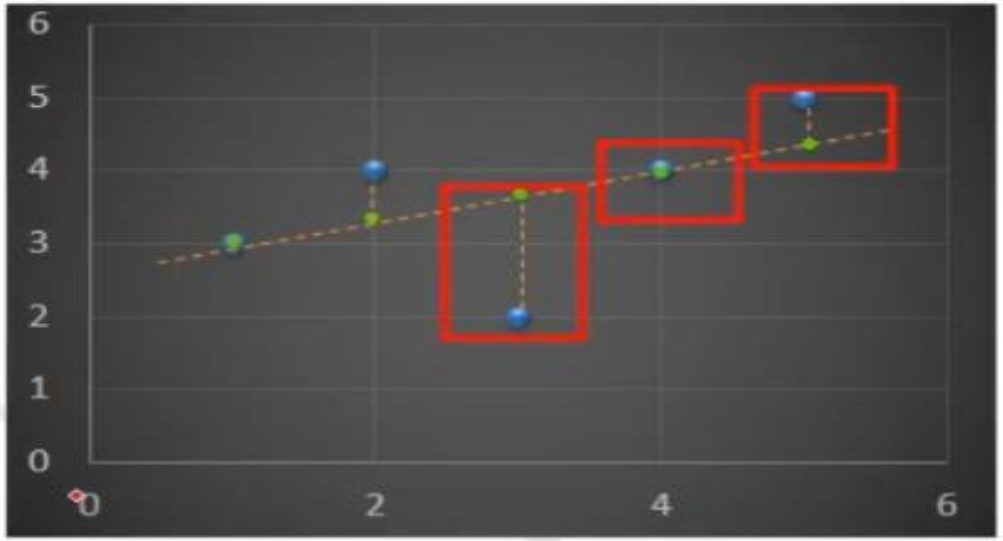
X	Y	Y'	Residual Error
1	3	3	3 – 3 = 0
2	4	3.2	
3	2	3.6	
4	4	4	
5	5	4.3	



Residual Error - Example

Consider an example below :

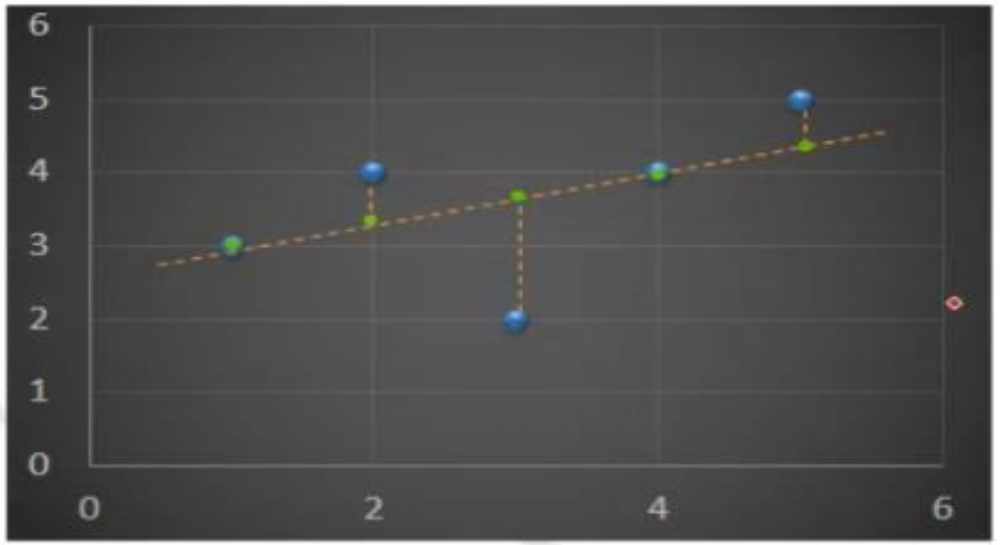
X	Y	Y'	Residual Error
1	3	2.8	3 – 2.8 = 0.2
2	4	3.2	4 – 3.2 = 0.8
3	2	3.6	2 – 3.6 = -1.6
4	4	4	4 – 4 = 0
5	5	4.3	5 – 4.3 = 0.7



Residual Error - Example

Consider an example below :

X	Y	Y'	Residual Error
1	3	2.8	3 – 2.8 = 0.2
2	4	3.2	4 – 3.2 = 0.8
3	2	3.6	2 – 3.6 = -1.6
4	4	4	4 – 4 = 0
5	5	4.3	5 – 4.3 = 0.7



We would want to find the line that has minimal squared residual error.

You might have noticed that in this example we have one independent and one dependent variable – SIMPLE LINEAR REGRESSION !

Simple Linear Regression

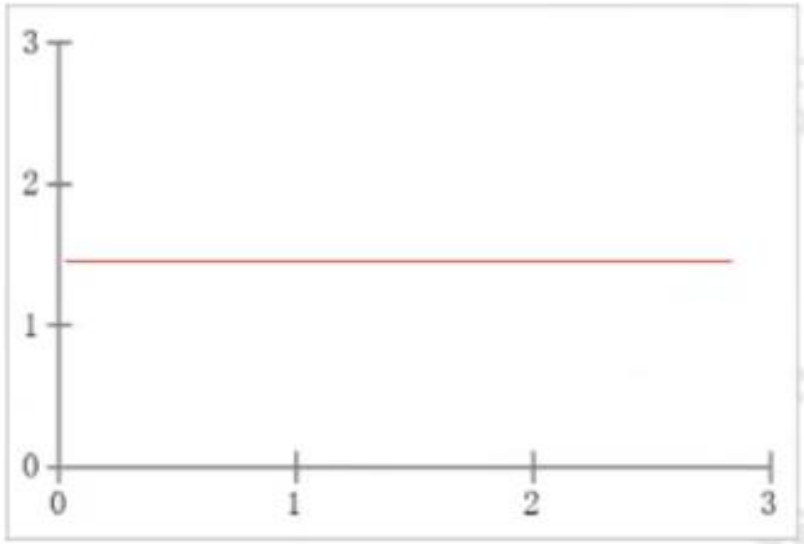
With simple linear regression, we have a single input (i.e. one independent variable).

$$Y_i = \beta_0 + \beta_1 * X_i$$

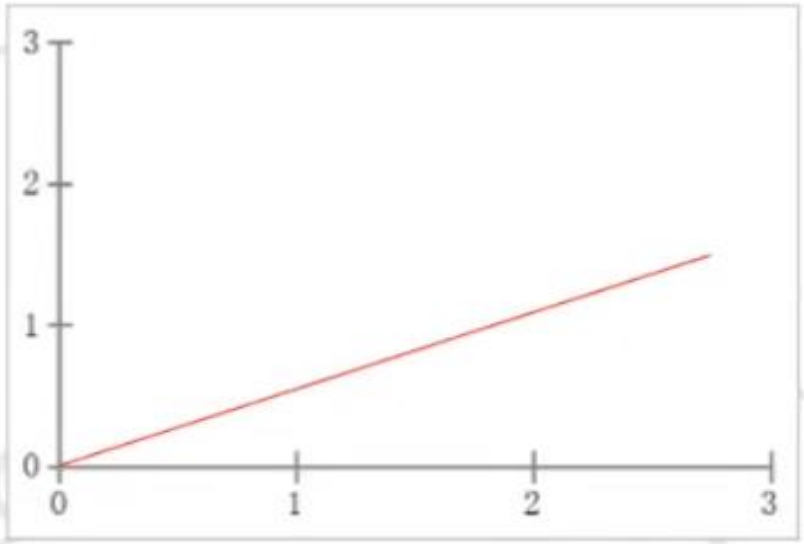
Linear Regression: Estimate the best values for β_0 & β_1

Parameters in Linear Regression

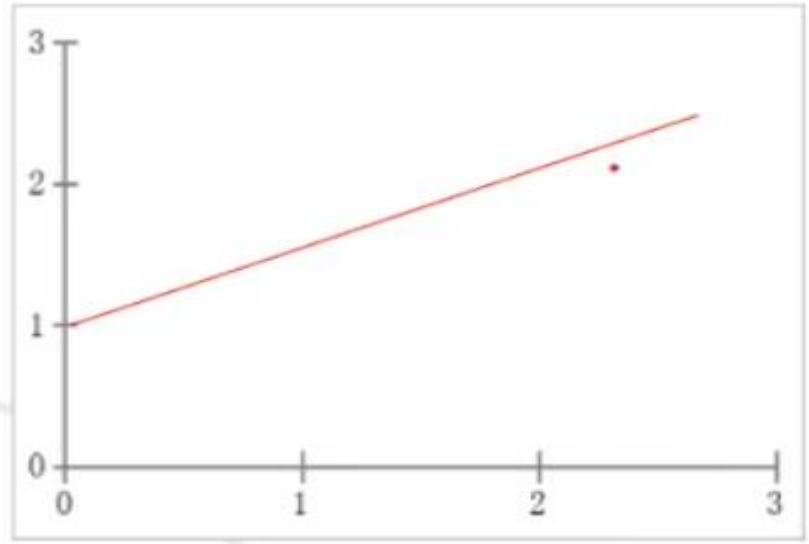
$$Y' = \beta_0 + \beta_1 * X$$



$\beta_0 = 1.5,$
$\beta_1 = 0$



$\beta_0 = 0,$
$\beta_1 = 0.5$



$\beta_0 = 1,$
$\beta_1 = 0.5$

Statistical Approach

When we have a single input, we can **use statistics to estimate the coefficients.**

$$Y_i' = \beta_0 + \beta_1 * X_i$$

- Y_i' → Best Approximation for Y_i
- β_0 → Intercept
- β_1 → Coefficient/Slope

Estimate the coefficients: Find β_0 & β_1 such that Errors are minimized.

Error Function/ Cost Function to be minimized: $\sum_{i=1}^{n}(Yi - Yi')^2$

Expected Output

Machine Learning Approach

Linear Regression is a machine learning algorithm based on **supervised learning**.

While training the model we are given :

x: input training data

y: labels to data

X	Y
1	3
2	4
3	2
4	4
5	5

Training phase: The model learns the best regression fit line by finding the best β_0 and β_1 values.

$\boldsymbol{\beta_0}$: intercept

$\boldsymbol{\beta_1}$: coefficient of x

Iterative Learning through Gradient Descent

Gradient Descent

- *Gradient Descent (GD)* is a generic optimization algorithm to find optimal solutions to many ML problems, e.g., Linear Regression, MLP, etc.

- The idea is to modify parameters iteratively in order to minimize a cost function.

Suppose one is lost in the mountains in a dense fog; He can only feel the slope of the ground below his feet. A good strategy to reach to the valley bottom quickly is to go downhill in the direction of the steepest slope.

Similarly, GD measures the local gradient (slope) of the error function w.r.t the parameter vector θ (in our case β_0 and β_1) and it goes in the direction of descending gradient.
- When the gradient is zero, minimum error is reached !

Gradient Descent

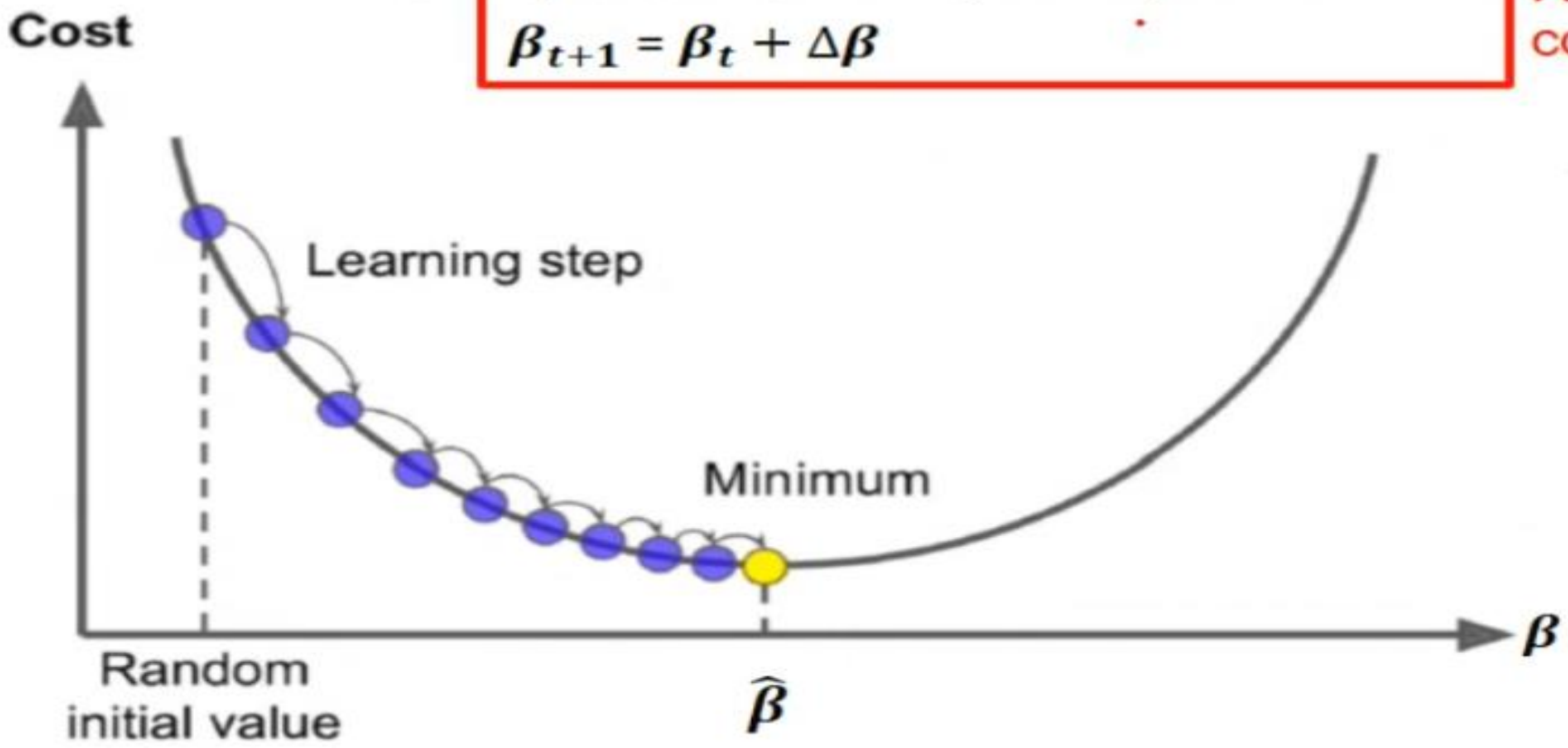
Start by initializing $\boldsymbol{\beta}$ with random values (called *random initialization*), and then improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm *converges* to a minimum.

$\boldsymbol{\beta_0}$ = r (random value)

$\Delta\boldsymbol{\beta} \propto \boldsymbol{Gradient\ of\ Cost\ function}$

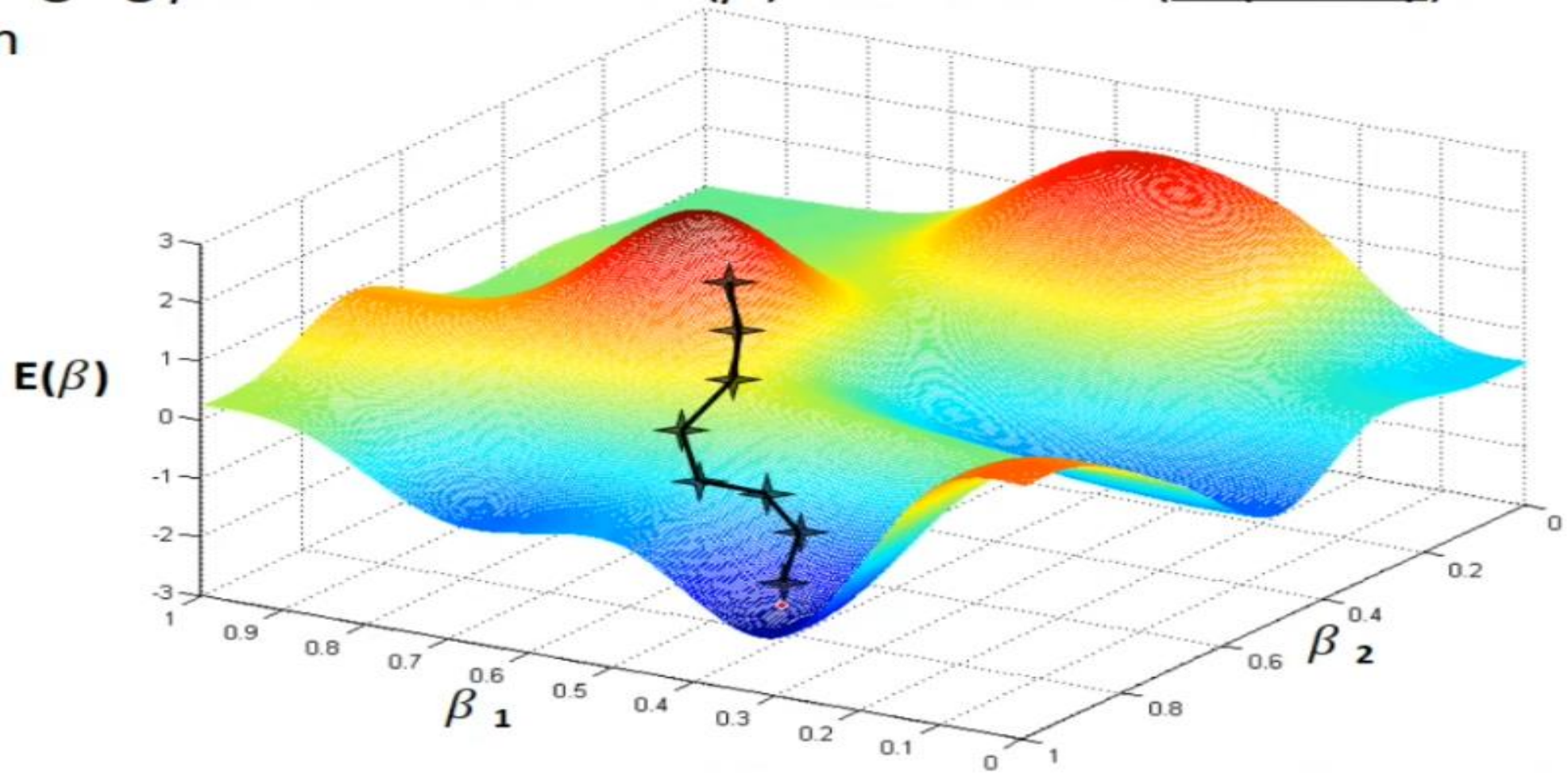
$\boldsymbol{\beta_{t+1}} = \boldsymbol{\beta_t} + \Delta\boldsymbol{\beta}$

Repeat until converges



Gradient Descent

- Start with initial values of β s
- Keep changing β s to reduce Error (β) until error is (hopefully) at a minimum



Gradient Descent

Goal: Minimize the cost function (Mean Squared Error). $S = \sum_{i=1}^{n}(Y_i - Y_i')^2$

Algo:

- Initialize β_0 and β_1 values randomly
- Unless the error value is acceptable OR there's no improvement in cost:
  Iteratively update these value in order to minimize the cost function

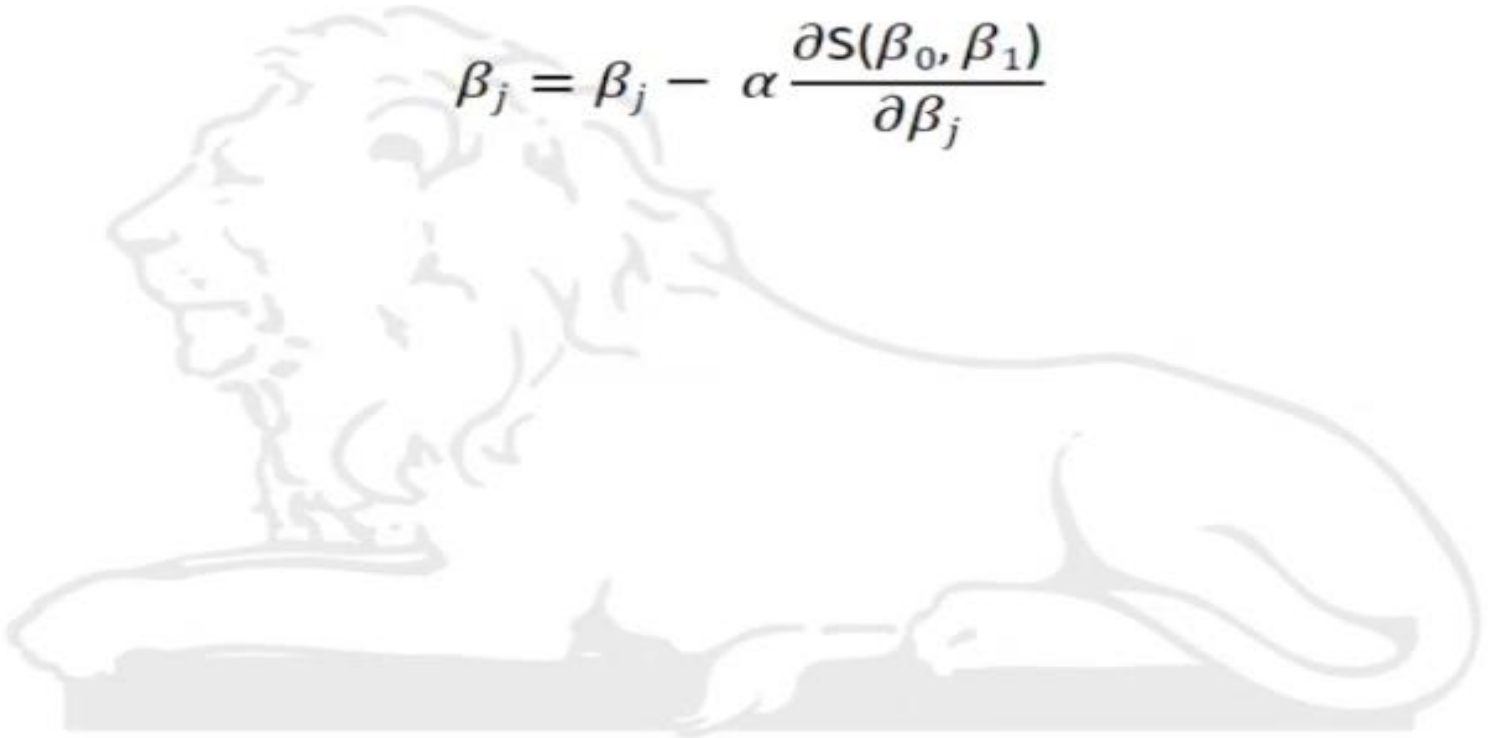
Iterative Approach

By the time model achieves the minimum cost function, it will have the best β_0 and β_1 values.

Gradient Descent - Update

Update : On every update, each coefficient is updated according to the following equation:

$$\beta_j = \beta_j - \alpha \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_j}$$

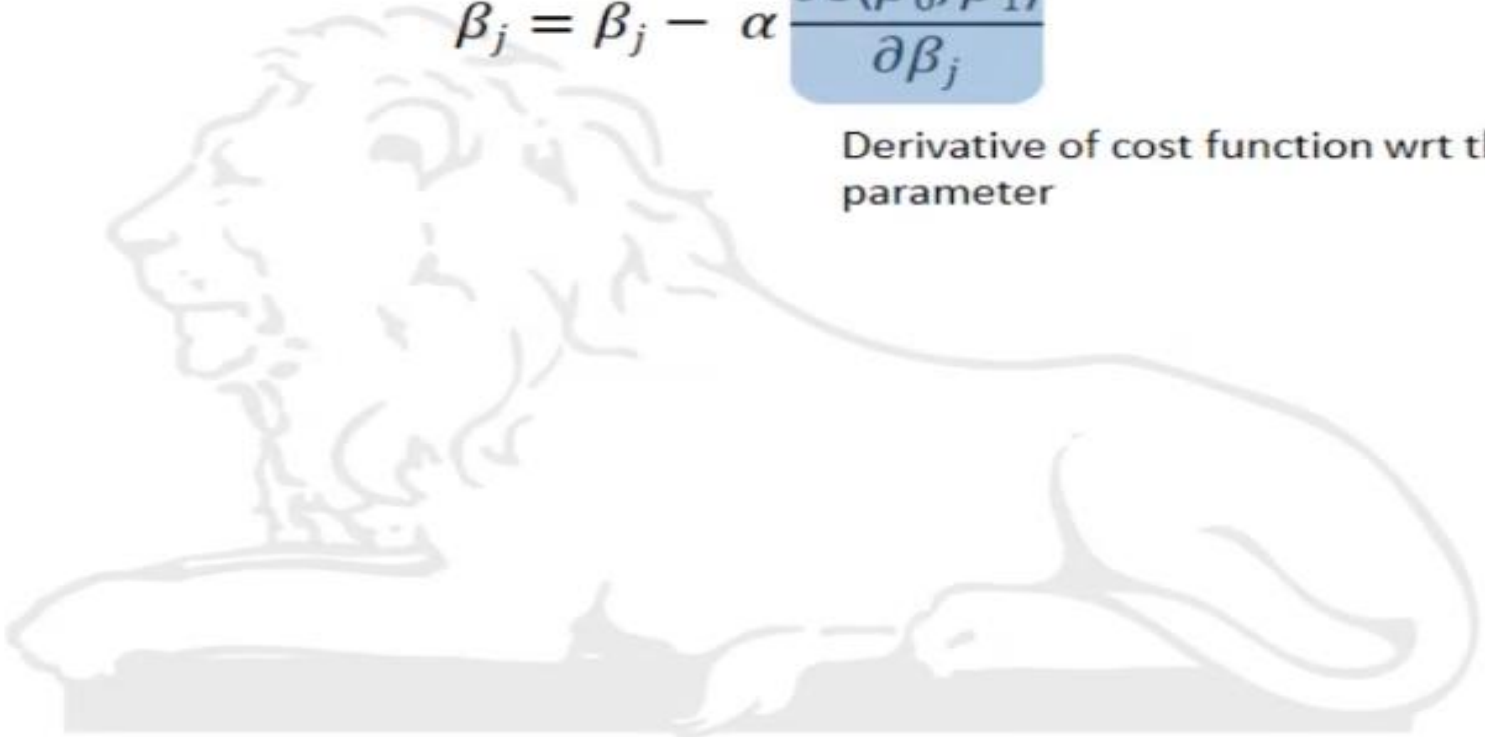


Gradient Descent - Update

Update : Each parameter/coefficient is updated according to the following equation:

$$\beta_j = \beta_j - \alpha \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_j}$$

Derivative of cost function wrt the parameter

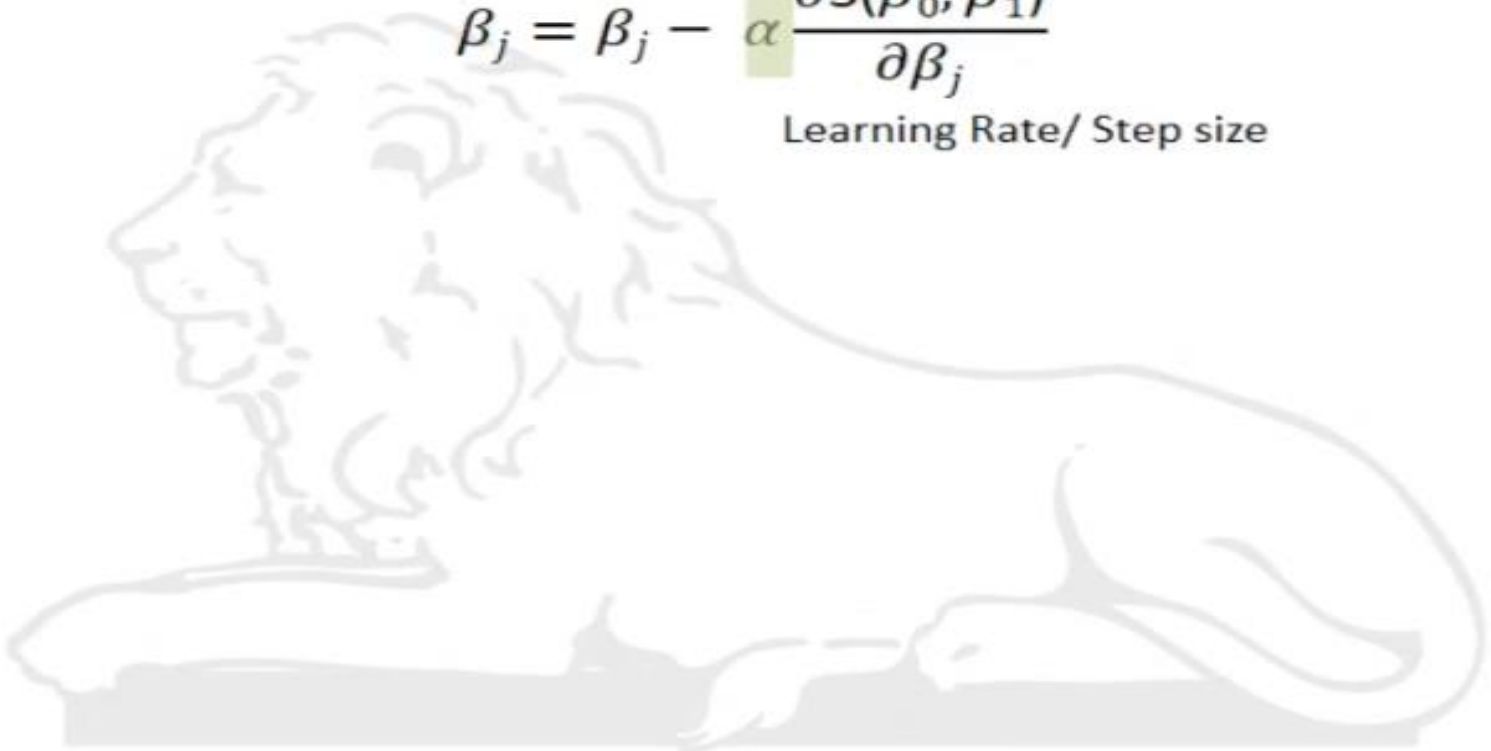


Gradient Descent - Update

Update : Each coefficient is updated according to the following equation:

$$\beta_j = \beta_j - \alpha \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_j}$$

Learning Rate/ Step size



Gradient Descent - Update

Update : Each coefficient is updated according to the following equation:

$$\beta_j = \beta_j - \alpha \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_j}$$

$$\frac{\partial S(\beta_0,\beta_1)}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i').\frac{\partial (Y_i - \beta_0 - \beta_1 X_i)}{\partial \beta_j}$$

Differentiating wrt β_1

$$\frac{\partial S(\beta_0, \beta_1)}{\partial \beta_1} = -\frac{1}{n}\sum_{i=1}^{n}[(Y_i - Y_i').X_i]$$

Gradient Descent - Update

Update : Each coefficient is updated according to the following equation:

$$\beta_j = \beta_j - \alpha \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_j}$$

$$\frac{\partial S(\beta_0,\beta_1)}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i').\frac{\partial (Y_i - \beta_0 - \beta_1 X_i)}{\partial \beta_j}$$

$$\frac{\partial S(\beta_0, \beta_1)}{\partial \beta_1} = -\frac{1}{n}\sum_{i=1}^{n}[(Y_i - Y_i').X_i] \qquad \frac{\partial S(\beta_0, \beta_1)}{\partial \beta_0} = -\frac{1}{n}\sum_{i=1}^{n}[(Y_i - Y_i')]$$

$$\beta_1 = \beta_1 + \frac{\alpha}{n}\sum_{i=1}^{n}[(Y_i - Y_i').X_i] \qquad \beta_0 = \beta_0 + \frac{\alpha}{n}\sum_{i=1}^{n}[(Y_i - Y_i')]$$

Gradient Descent - Example

Update : Each coefficient is updated according to the following equation:

$$\beta_j = \beta_j + \frac{\alpha}{n}\sum_{i=1}^{n}[(Y_i - Y_i').X_j^i]$$

Let us start with some random values & see how we can learn the optimal values of β_0, β_1

Bringing back our toy example

X	Y
1	3
2	4
3	2
4	4
5	5