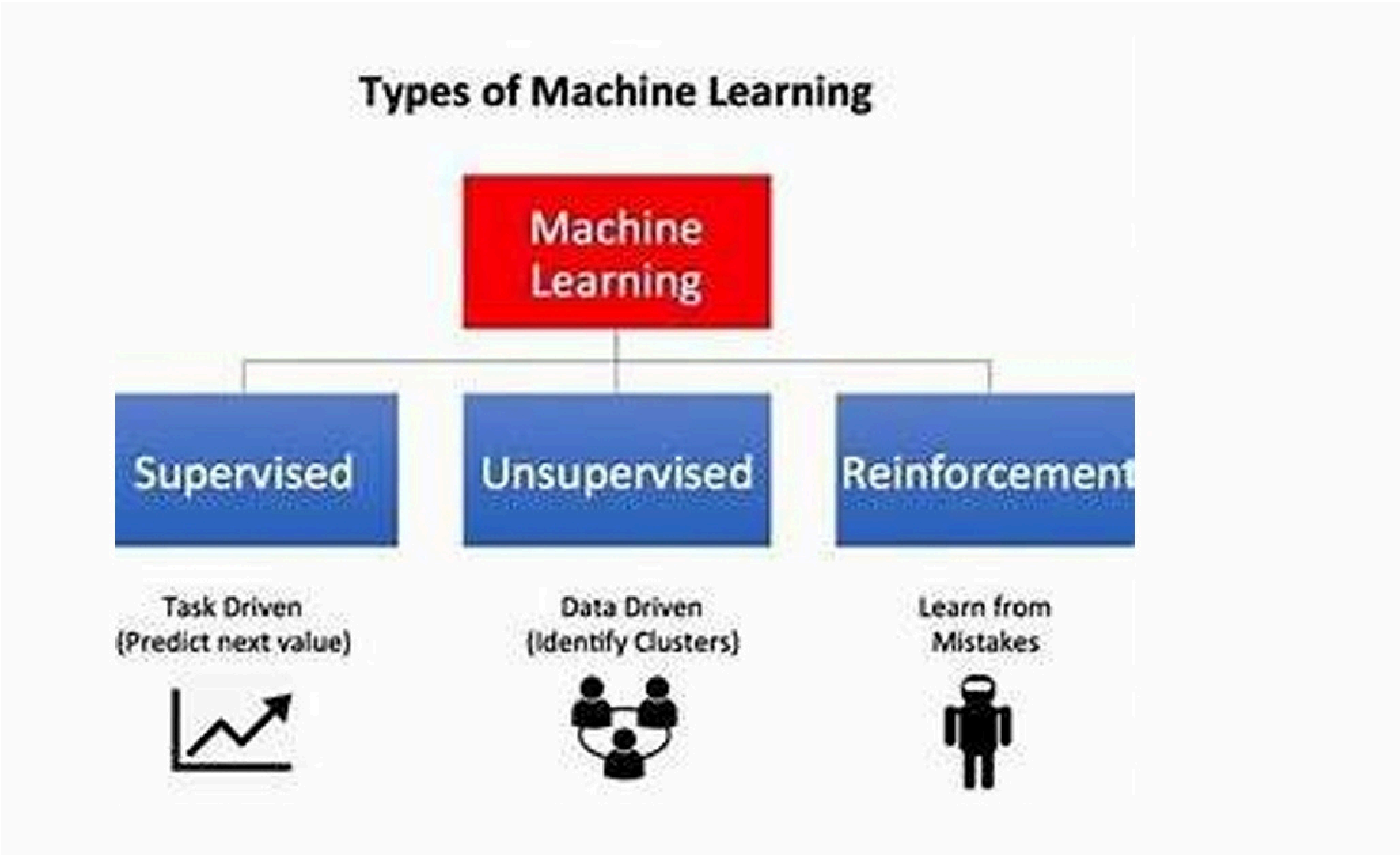
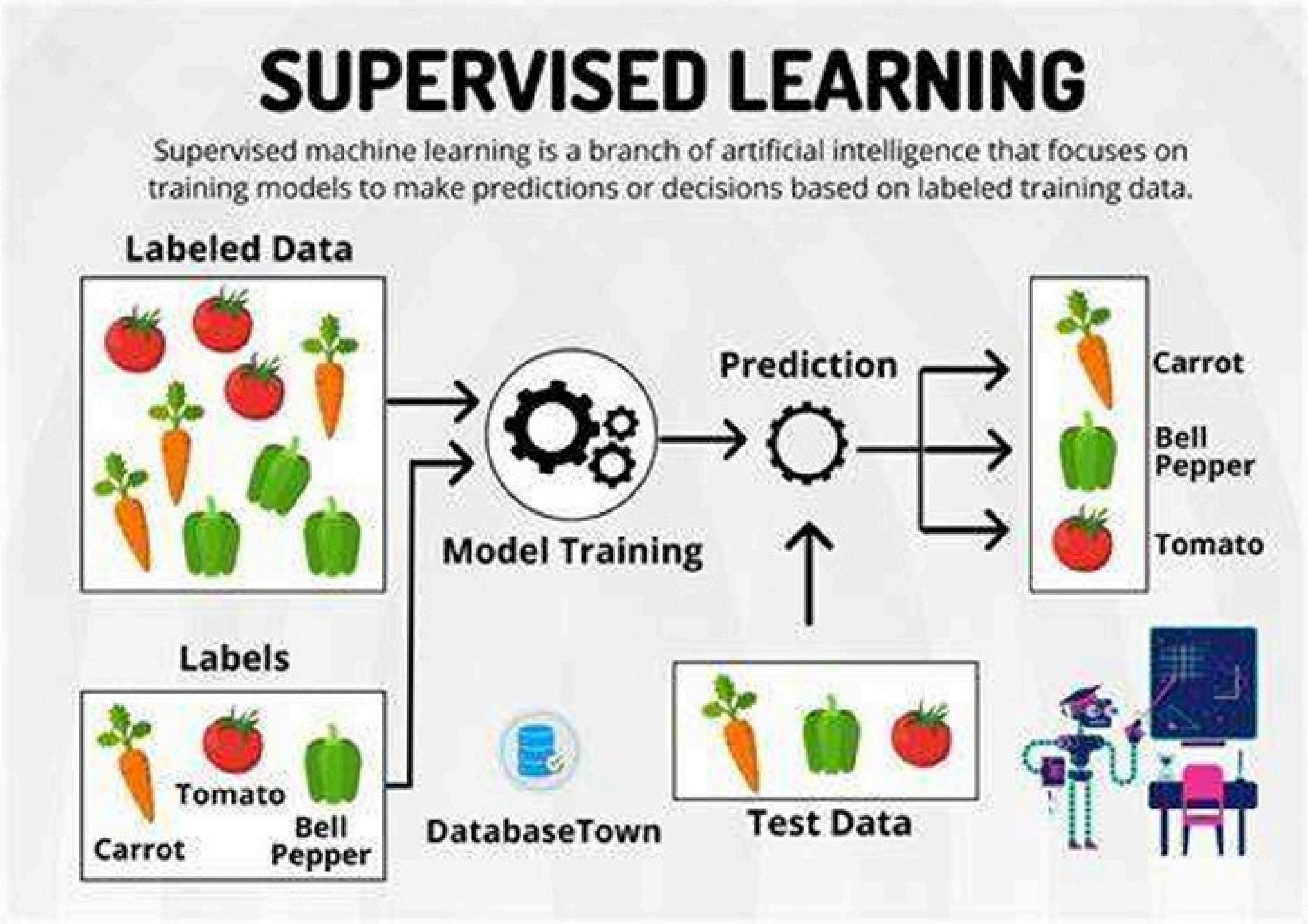

Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task Driven
(Predict next value)
Data Driven
(Identify Clusters)
Learn from
Mistakes

Supervised Learning

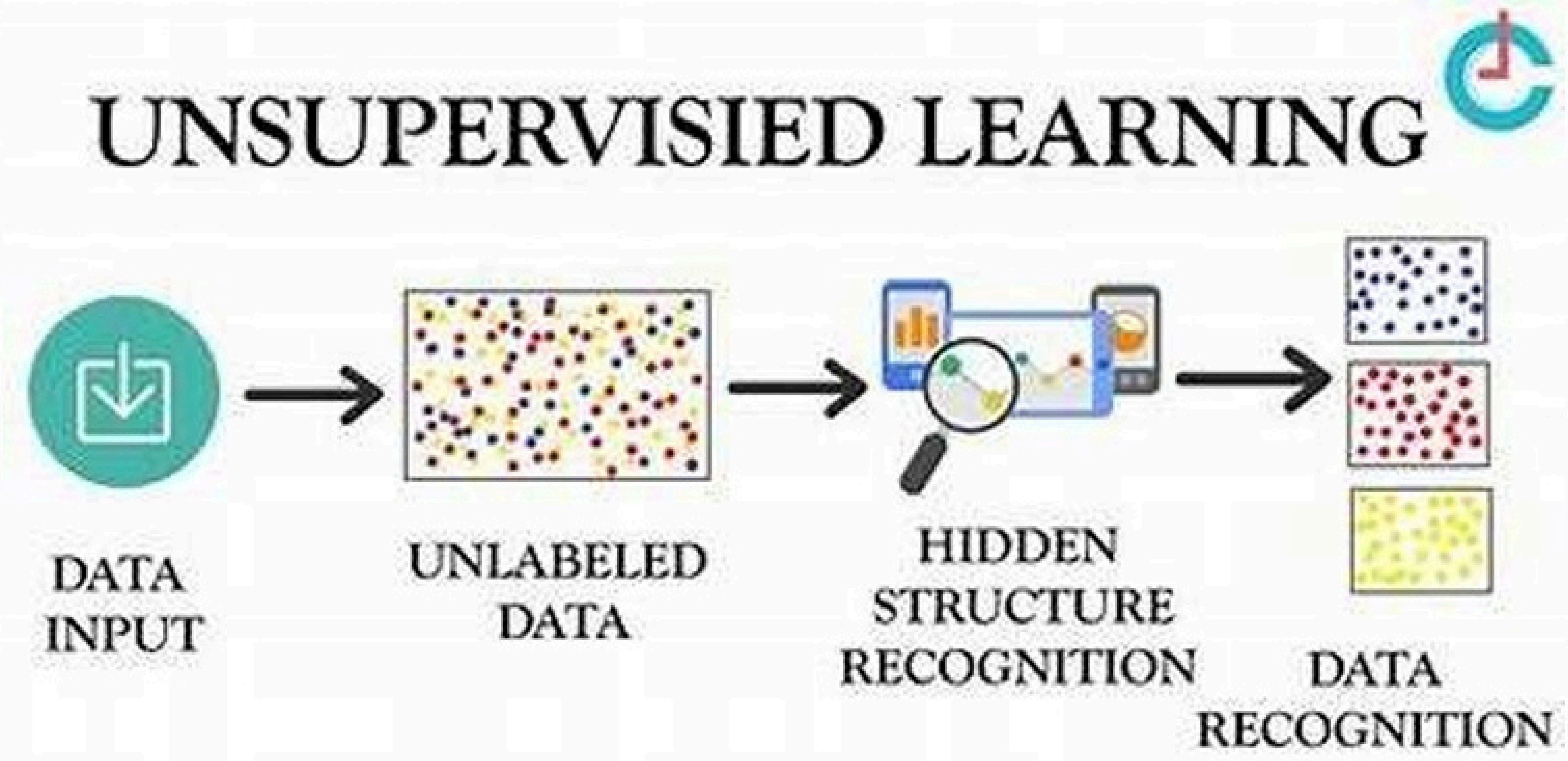


Supervised Learning Algorithm Types

1.Regression
Linear regression
Polynomial regression
Naïve Bayes
Gradient Boosting
Lasso regression
Ridge regression
Elastic regression

2.Classification
Logistic regression
Support vector machine
Decision tree
Random forest
K nearest neighbors

Unsupervised Learning Algorithm



Unsupervised Learning Algorithm

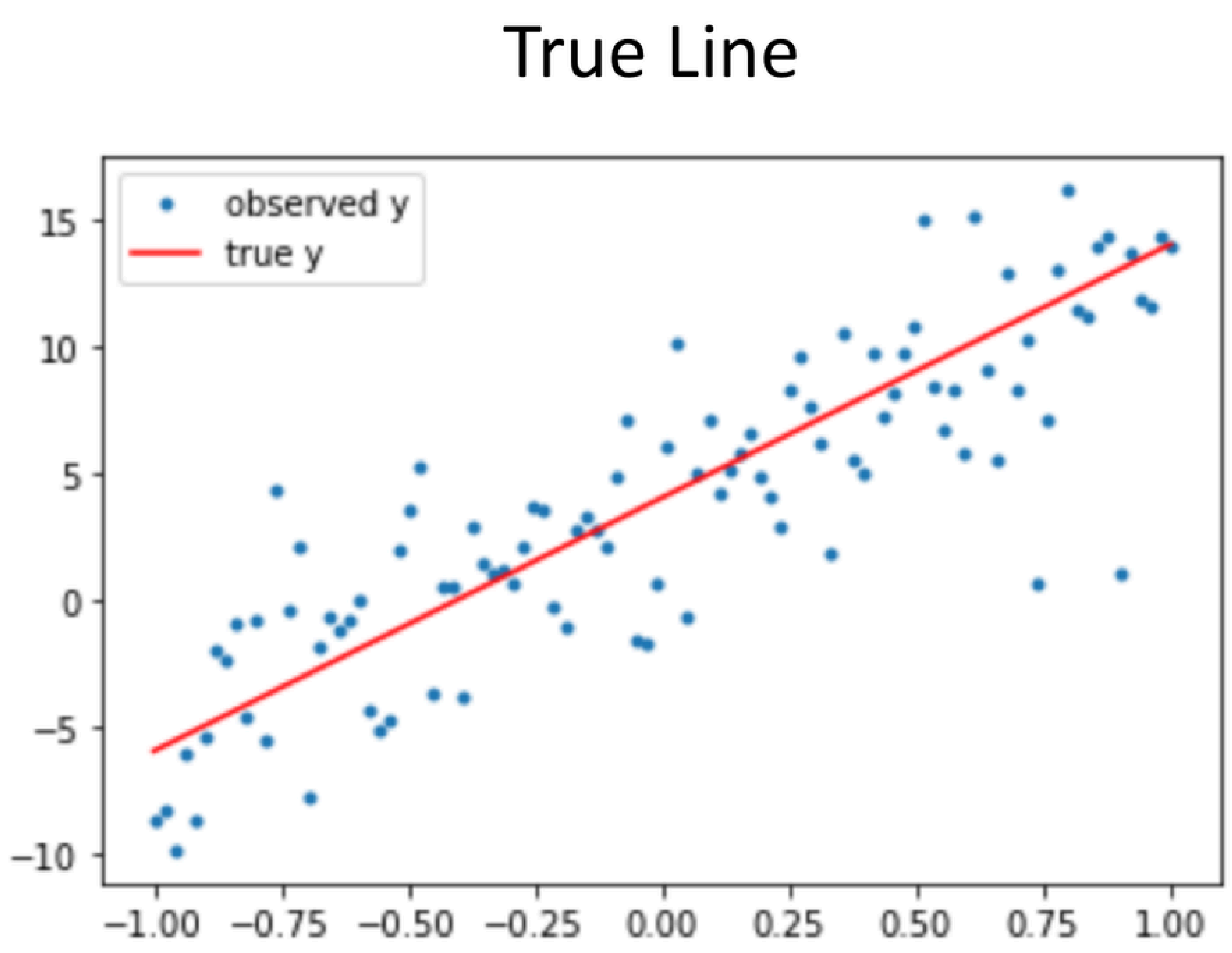
- Basic concept of unsupervised learning
- Types: Clustering, Dimensionality Reduction, Association Rule Learning
- K-Means Clustering
- Hierarchical Clustering: Agglomerative, Divisive
- Density-based Clustering: DBSCAN
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA

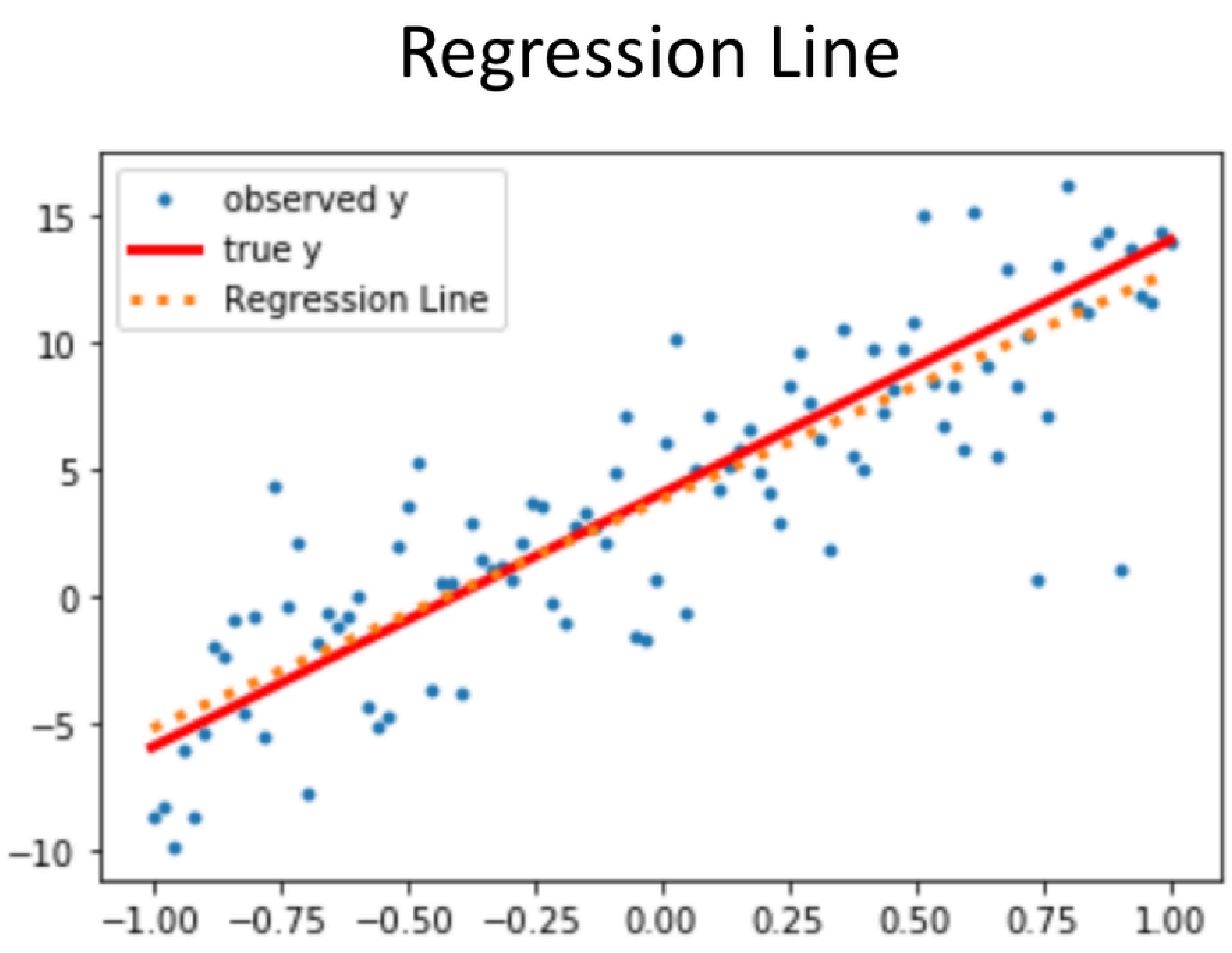
Reinforcement Learning

REINFORCEMENT LEARNING AGENT
EPISODE: 4502
DELL

CGPA (out of 4)	Package (LPA)
4.0	45
3.8	35
3.6	28
3.4	22
3.2	18
3.0	15
2.8	12
2.6	10

CGPA (out of 4)	Internship Score (out of 100)	Projects Completed	Package (LPA)
4.0	95	5	50
3.8	90	4	40
3.6	85	3	35
3.4	80	3	28
3.2	75	2	22
3.0	70	2	18





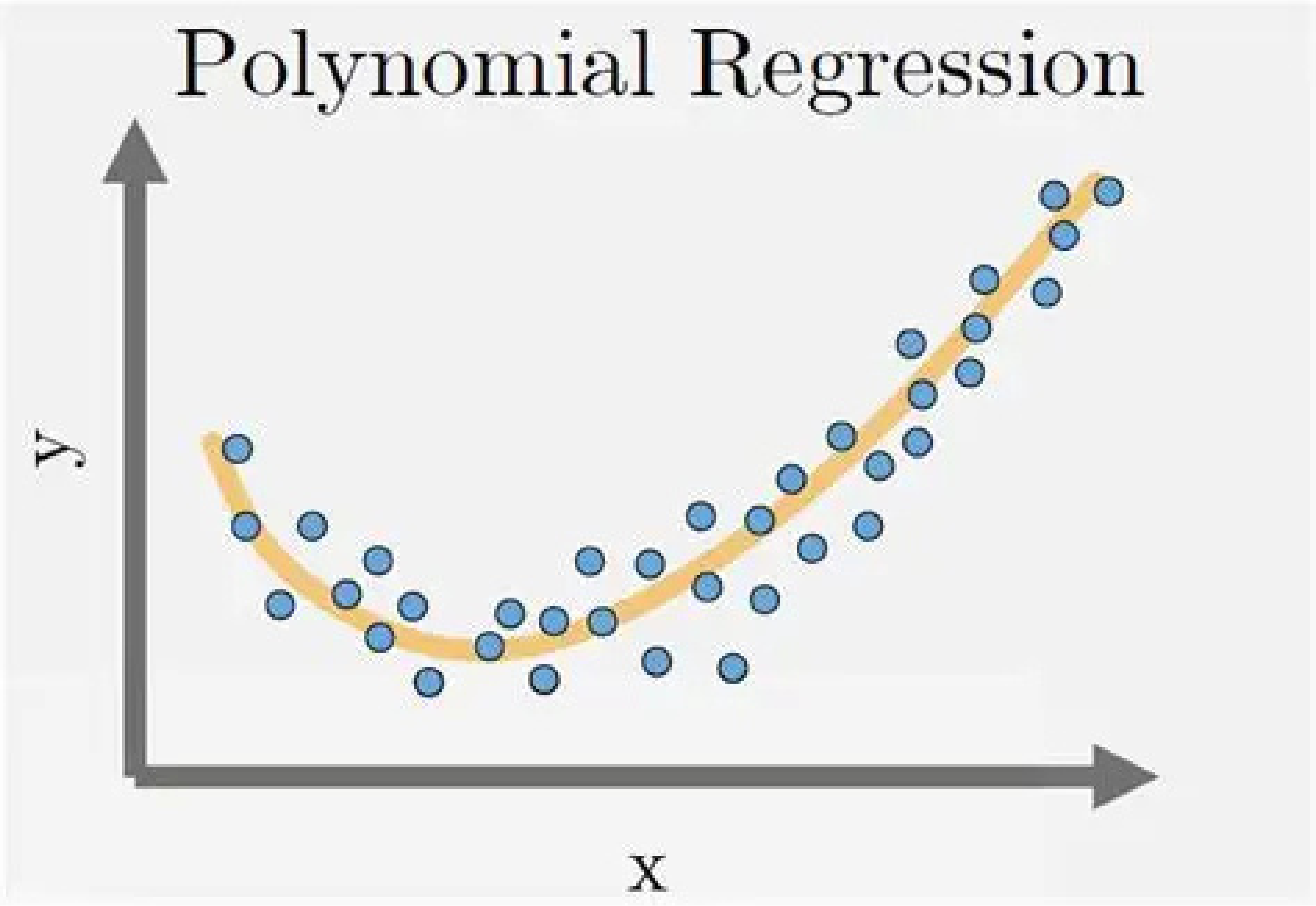
$$y = mx + b$$

Let's break it **clearly**:

---

1. Components of the Equation

- y → **Dependent variable** (what you want to predict, e.g., Package)
- x → **Independent variable** (the predictor, e.g., CGPA)
- m → **Slope / Coefficient** (how much y changes for a unit change in x)
- b → **Intercept** (value of y when x = 0)


Polynomial Regression
y
x

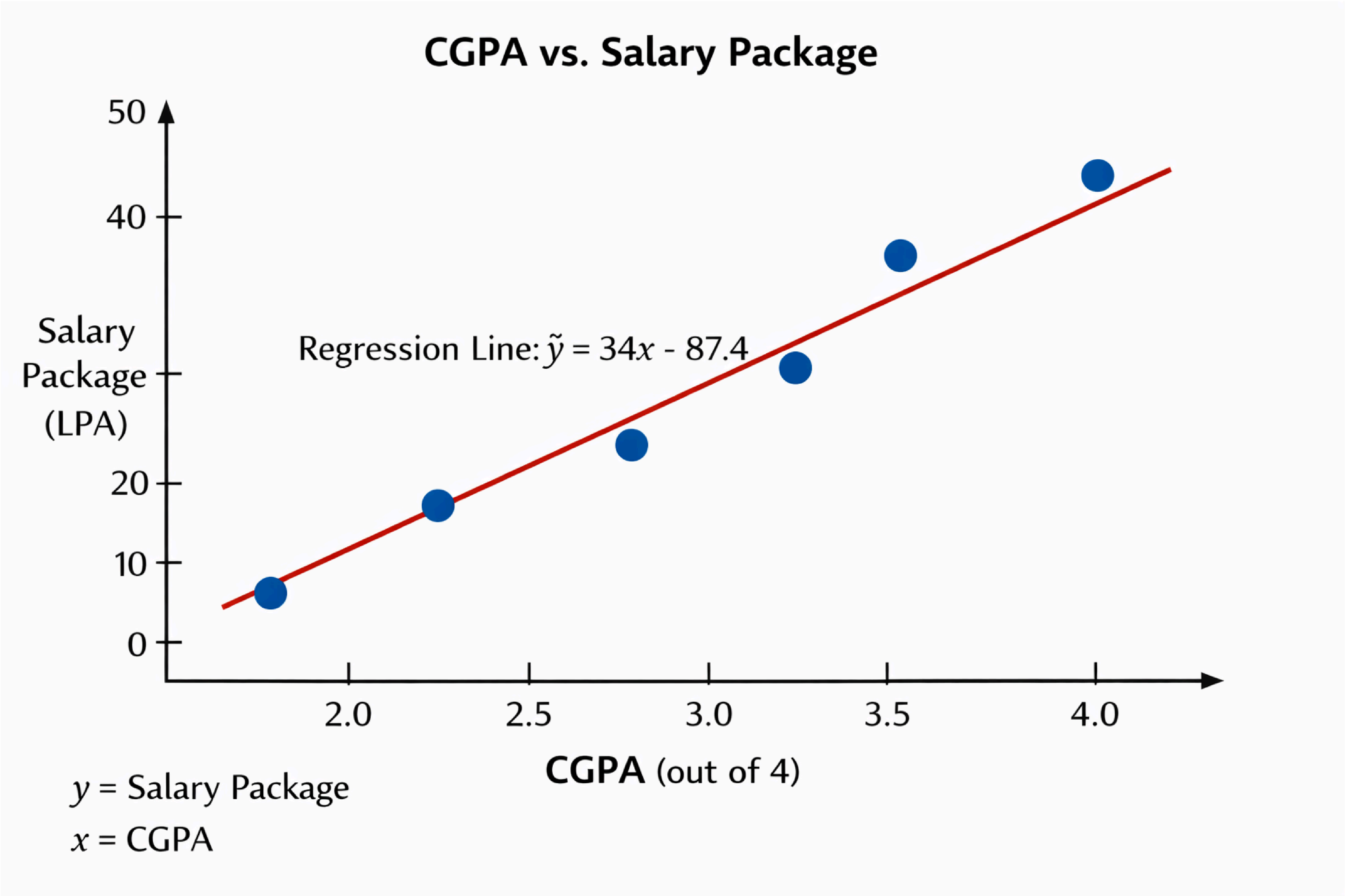
Problem Statement (Simple Linear Regression)

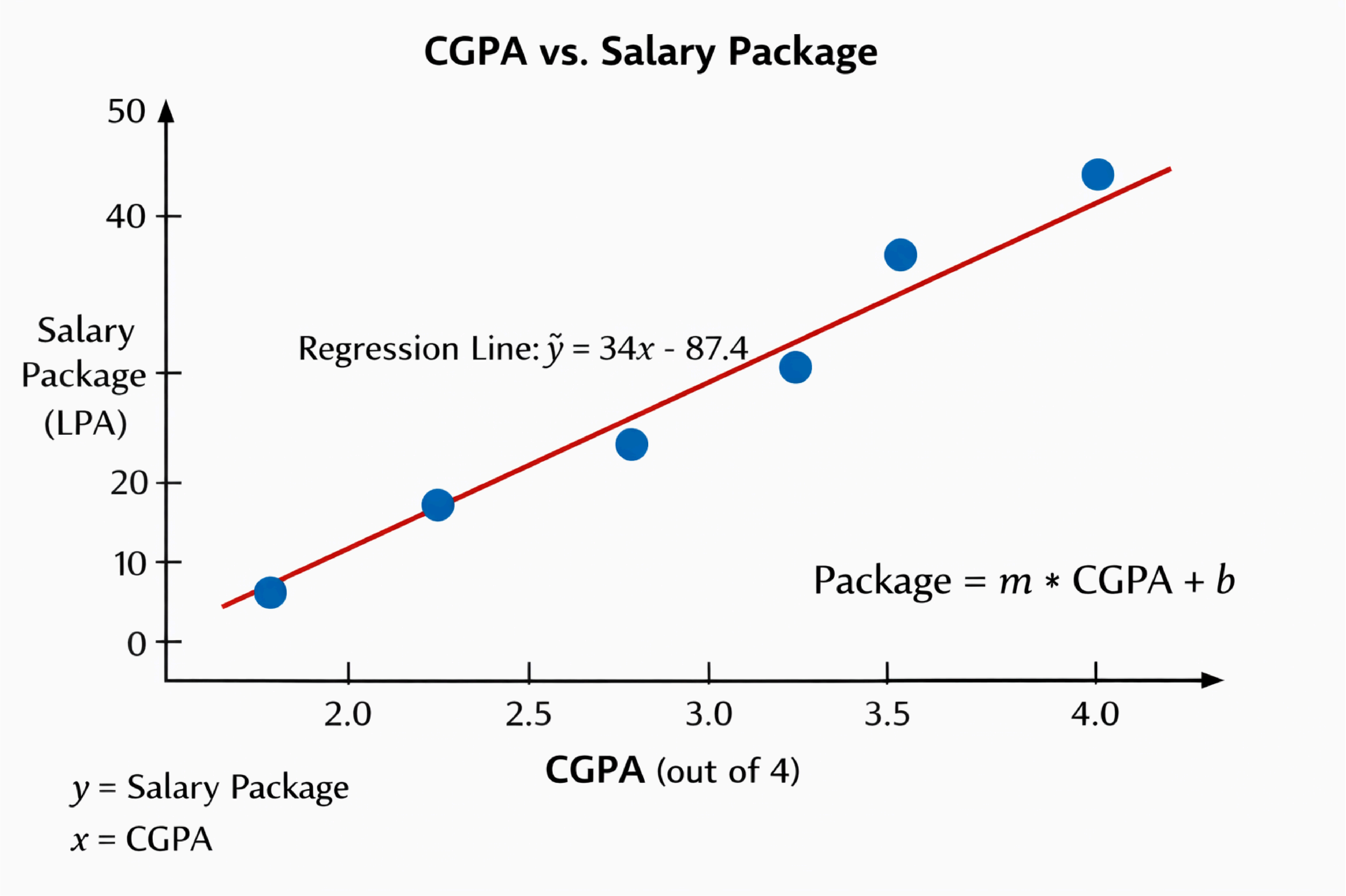
Suppose we have a dataset of **CGPA (x)** vs **Package in LPA (y)**:

CGPA (x)	Package (y)
4.0	50
3.8	40
3.6	35
3.4	28
3.2	22

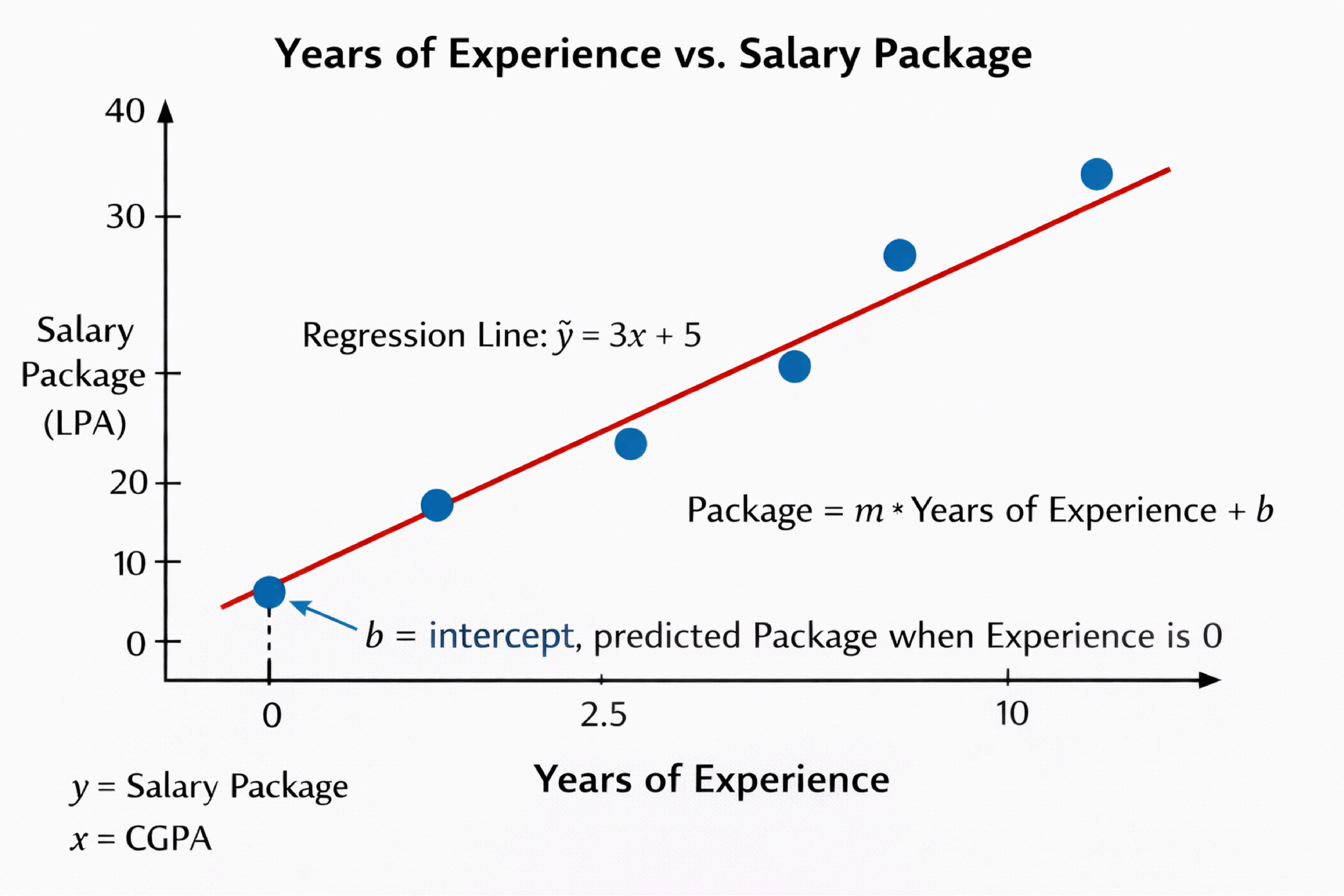
We want to find the regression line:

$$\hat{y} = mx + b$$


CGPA vs. Salary Package
50
40
Salary
Package
(LPA)
20
10
0
Regression Line: $\tilde{y} = 34x - 87.4$
2.0
2.5
3.0
3.5
4.0
CGPA (out of 4)
y = Salary Package
x = CGPA



CGPA vs. Salary Package
50
40
20
10
0
Salary Package (LPA)
Regression Line: $\tilde{y} = 34x - 87.4$
Package = m * CGPA + b
2.0
2.5
3.0
3.5
4.0
CGPA (out of 4)
y = Salary Package
x = CGPA



Years of Experience vs. Salary Package
40
30
20
10
0
Salary
Package
(LPA)
Regression Line: $\tilde{y} = 3x + 5$
Package = m * Years of Experience + b
b = intercept, predicted Package when Experience is 0
0
2.5
10
Years of Experience
y = Salary Package
x = CGPA

for linear regression we use a ordinary least square method

When we calculate m (slope) and b (intercept) for **training a linear regression model**, we are trying to **find the best-fit line** for all the training data points.

To do this, we **use a loss function** (Mean Squared Error, MSE) that measures **how far the predicted values are from the actual values**:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^2$$

- x_i → input feature
- y_i → actual output
- $\hat{y}_i = mx_i + b$ → predicted output

Important:

- We **don't update the line for one data point at a time.**
- Instead, we **calculate the slope m and intercept b for all training data at once** by minimizing the **total loss.**
- After training, the line $y = mx + b$ is the **best-fit line** for the training data.

When we **calculate the training loss**, we **only use the actual values from the training data** and the **predicted values from the line we are fitting.**

- **Actual values** → y_i from training dataset
- **Predicted values** → $\hat{y}_i = mx_i + b$ from the current line

lEAST SQUAREA METHOD

1 Slope m

$$\boxed{m = \frac{n\sum(x_iy_i) - (\sum x_i)(\sum y_i)}{n\sum(x_i^2) - (\sum x_i)^2}}$$

Where:

- n = number of training points
- x_i = input feature values
- y_i = actual output values

2 Intercept b

$$\boxed{b = \frac{\sum y_i - m\sum x_i}{n}}$$

Also we have gradient descent
in gradient descent we test data one by one and update the m and b but in least square we calculate m and b at one

we use gradient descent in deep learning neural network