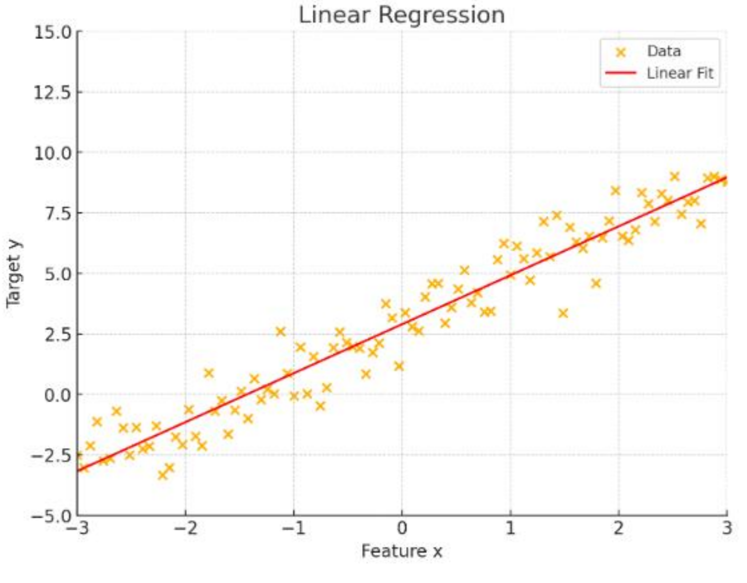
1. Linear Regression :

Linear regression models the relationship between a **dependent variable (Y)** and **independent variable(s) (X)** by fitting a linear equation.

$$Y = \beta 0 + \beta 1 * X1 + \beta 2 * X2 + ... + \beta n * Xn + \varepsilon$$

- y = predicted value
- β0 = intercept
- β1, β2, ..., βn = coefficients
- x1, x2, ..., xn = input features
- ε = error term



$$y = \beta 0 + \beta 1 * x + \varepsilon$$

- `y`: The **predicted value** (target variable).
- `x`: The **input feature** (independent variable).
- `β0`: The **intercept** — the value of `y` when `x = 0`.
- `β1`: The **slope** — how much `y` increases for every unit increase in `x`.
- `ε`: The **error term** — accounts for randomness or noise not captured by the model.

the plot shows:

- **Blue dots**: Actual data points — (x, y_true)
- **Red line**: The best-fit line found by minimizing the **Sum of Squared Errors (SSE)**:

$SSE = \Sigma (y_i - (\beta 0 + \beta 1 * x_i))^2$

The algorithm finds β0 and β1 such that the line passes as close as possible to all points.

✅ Use Cases:

- Predicting house prices
- Forecasting sales
- Estimating salary based on experience

📌 Importance:

Simple, fast, interpretable, great for baseline models.

✅ **Use Linear Regression when**:

- The relationship between input and output is **linear**.
- You are predicting a **continuous variable** (e.g., house price, salary).

2. **Logistic Regression**

Logistic Regression is used for **classification tasks**. It predicts the **probability** that a given input belongs to a particular category. It uses the **sigmoid (logistic) function** to squash output between 0 and 1.

Formula (for binary classification):

$$P(y = 1 \mid x) = 1 / (1 + e^{-(\beta 0 + \beta 1 * x1 + \beta 2 * x2 + ... + \beta n * xn)})$$

`β0` **Intercept** — shifts the curve left or right

`β1` **Weight (coefficient)** — controls the steepness of the curve

`e` Euler's number (≈ 2.718), the base of the natural logarithm

`e^-(β0 + β1*x)` Part of the **sigmoid function**, helps map any number into (0, 1) range

What Logistic Regression Does:

- It **does not predict a value directly**, but a **probability**.
- The curve is called a **sigmoid function** (S-shaped).
- The output is always between **0 and 1**, which makes it perfect for **binary classification**.

✅ Use Logistic Regression When:

- The output is **binary** (yes/no, spam/ham, pass/fail, etc.).
- You want **probability estimates**, not just class labels.

- The relationship between `x` and `P(y=1)` is **non-linear** in shape (S-curve).

Use Case:

- Spam detection (spam or not spam)
- Disease prediction (yes or no)
- Credit risk classification (default or no default)

3. Polynomial Regression

Polynomial Regression is an extension of Linear Regression where we include **polynomial terms** (e.g., x^2, x^3). It helps model **non-linear relationships**.

y = β0 + β1*x + β2*x^2 + ε

Use Case:

- Modeling curved trends (e.g., growth curves)
- When linear regression underfits due to non-linearity

No regularization means you're **not adding any penalty** to the cost function that the model is trying to minimize. You're purely minimizing the error between your predictions and the actual target values.

4. Ridge Regression (L2 Regularization)

Ridge Regression is a type of linear regression that includes **L2 regularization** — it adds a **penalty term** proportional to the **square of the coefficients** to reduce overfitting and multicollinearity.

Minimize: Sum of Squared Errors + λ * (β1^2 + β2^2 + ... + βn^2)

Use Case:

- When you want to keep all features but reduce model complexity
- When features are highly correlated
- Adds a squared penalty on the weights.
- Helps in reducing overfitting by shrinking large weights.

5. Lasso Regression (L1 Regularization)

Lasso Regression adds **L1 regularization** — a penalty equal to the **absolute values of coefficients**. It can **shrink some coefficients to zero**, effectively performing **feature selection**.

Minimize: Sum of Squared Errors + λ * (|β1| + |β2| + ... + |βn|)

- Adds a squared penalty on the weights.

- Helps in reducing overfitting by shrinking large weights.

- Useful for **feature selection**.
- Can completely remove irrelevant features.
- **Fitted Model**: May ignore less important features.
- **Cost Function**: Diamond-shaped contours — sharp corners push weights to zero.

Use Case:

- When you want a **sparse model** (few features)
- For feature selection in high-dimensional datasets

6.Elastic Net Regression (L1 + L2)

Elastic Net Regression combines both **L1 and L2 regularization**. It balances the benefits of **Ridge (shrinkage)** and **Lasso (feature selection)**, especially helpful when predictors are correlated.

Cost Function:

Minimize: Sum of Squared Errors + λ1 * (|β1| + ... + |βn|) + λ2 * (β1^2 + ... + βn^2)

- Mixes both penalties to get the best of both worlds.
- Controlled using `l1_ratio` parameter.
- **Fitted Model**: Balanced — selects features and avoids overfitting.
- **Cost Function**: Mix of round (L2) and sharp (L1) shapes in contours.

Use Case:

- When you have **many correlated features**
- When you want both regularization and feature selection

Regression Type	Target	Regularization	Cost Function Type	Feature Selection	Use Case
Linear	Continuous	❌ No	MSE	❌	Basic trends
Polynomial	Continuous	❌ No	MSE	❌	Complex curves
Ridge (L2)	Continuous	✅ L2	MSE + θ^2	❌	Reduce overfitting
Lasso (L1)	Continuous	✅ L1	MSE +	θ	
Elastic Net	Continuous	✅ L1 + L2	MSE +	θ	+ θ^2
Logistic	Categorical	Optional (L1/L2)	Log Loss	✅ (if regularized)	Classification tasks