Unit-1

Linear Methods for Regression and Classification:

- Classification is the process of predicting the class labels for the input data to categorize the data into classes.
- This can be done by building models. A classification model acts as a decision tree which separates the provided input data and gives each group of data (class) a label.
- These models are derived and can be built by providing labeled training data.

Hence, classification models are used to predict class labels for new, unlabeled data.

Ex: in email spam detection, a classification model can be trained on labeled emails (spam or not spam) to predict the class of incoming emails.

- Regression, on the other hand, is the process of predicting continuous numerical values based on the provided input features.
- It aims to find a mathematical relationship between the input variables and the output variable.

Ex: in house price prediction, regression can be used to build a model that predicts the price of a house based on factors like its area, number of rooms, location, etc.

Supervised Learning:

It is when the labeled data sets are provided as training data. The class labels for the new instances are predicted from that training labeled data.

Workflow in Supervised Learning

- **Data Collection and Preparation**: Gather and preprocess data, including labeling examples, cleaning, normalizing, and splitting into training, validation, and test sets.
- **Model Training**: Use the training set to help the model learn by iteratively adjusting parameters to minimize errors.
- **Model Evaluation**: Assess model performance on a validation set to fine-tune hyperparameters and on a test set to evaluate generalization.
- **Deployment**: Once validated, the model is deployed for real-world use, with regular monitoring for accuracy.

Common Algorithms

- **Linear Regression**: For regression tasks, fits a linear relationship between input variables and the output.
- **Logistic Regression**: For binary classification tasks, estimates probabilities to classify inputs into two categories.

- **Decision Trees and Random Forests**: Nonlinear models that are highly interpretable and suitable for both classification and regression.
- **Support Vector Machines (SVM)**: Powerful for classification by finding the optimal boundary between classes.
- **K-Nearest Neighbors (KNN)**: Classifies data points based on the majority class among the closest neighbors.
- **Neural Networks**: Highly flexible, able to approximate complex functions, suitable for tasks requiring high-dimensional data, such as image recognition.

Variable Types:

In machine learning (ML), various variable types play distinct roles in model building. Understanding these variable types is crucial because they influence how the data is handled, preprocessed, and used in training the model.

1. Input Variables (Features):

- **Definition**: These are the variables used as input to the model, typically representing the characteristics or attributes of the data you're trying to analyze or predict.
- These are also known as **Independent Variables** or **Predictors(**used to predict the dependent variable (target)) or **Features**.
- **Types**:
    - **Numerical**: Variables that can take any numeric value within a range. E.g., height, weight, temperature, age.
        - It can be **qualitative** or **quantitative**
    - **Categorical**: Variables that represent categories or labels. E.g., colors, types of animals, or countries.
        - **Nominal**: Categories with no inherent order (e.g., red, blue, green).
        - **Ordinal**: Categories with a meaningful order (e.g., low, medium, high).

2. Target Variables (Labels or Output):

- **Definition**: These are the variables that the model is trying to predict or classify. The target can either be a **single value** or a **set of values**
- These are also known as **Dependent Variables** (dependent on the independent variables and represents the outcome you're trying to predict.)
- **Types**:
    - **Regression**: When the target variable is continuous (e.g., predicting house prices, stock prices).

    - **Classification**: When the target variable is categorical (e.g., binary classification like "spam" or "not spam", or multi-class classification like identifying types of flowers).
        - This can be binary too - These are variables with only two possible outcomes, usually represented as 0 and 1.
          Example: Whether a customer purchased a product or not (1 = Yes, 0 = No).

3. Hyperparameters:

- **Definition**: These are variables that influence the learning process and structure of the model, but aren't learned from the data itself. They control aspects such as the complexity of the model, the learning rate, or the number of iterations.
- **Examples**:
    - Learning rate in gradient descent.
    - Number of layers in a neural network.
    - Regularization strength in regression models.

Difference B/W Classification and Regression:

Classification and Regression are both the same supervised learning tasks used to predict the target variable based on existing sample instances. The main difference lies in the **nature of the target variable** (the output you're trying to predict).

Key Differences Between Classification and Regression:

1. **Target Variable Type**:
    - **Classification**: The target variable is **categorical**. The model's goal is to assign input instances to one of several predefined classes or categories.
        - **Example**: Predicting whether an email is spam or not (binary classification), or predicting the species of a flower (multi-class classification).
    - **Regression**: The target variable is **continuous**. The model's goal is to predict a numerical value.
        - **Example**: Predicting house prices, predicting the temperature on a given day.
2. **Output**:
    - **Classification**: The model produces a class label or probability of belonging to a specific class.
        - **Example**: In a binary classification task, the output could be "0" (not spam) or "1" (spam).
    - **Regression**: The model produces a continuous numeric output.

        - **Example**: In a regression task, the output could be a specific value like 450,000 for a house price.

3. **Evaluation Metrics**:
    - **Classification**: The evaluation metrics focus on the accuracy of the model in assigning the correct class. Common metrics include:
        - Accuracy
        - Precision, Recall, F1-score
        - Confusion matrix
        - ROC-AUC
    - **Regression**: The evaluation metrics focus on how close the predicted values are to the actual values. Common metrics include:
        - Mean Absolute Error (MAE)
        - Mean Squared Error (MSE)
        - Root Mean Squared Error (RMSE)
        - R-squared (R^2)

Linear Regression Model

Linear Regression is one of the simplest and most commonly used supervised learning algorithms for **predicting a continuous output (numerical value)** based on input features. It establishes a **linear relationship** between the dependent variable (target) and one or more independent variables (features).

1. Mathematical Equation of Linear Regression

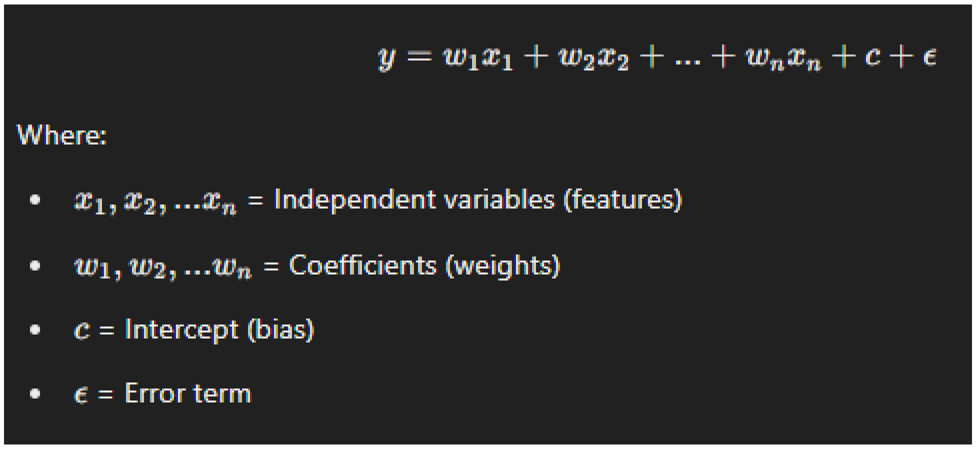
For a **single** independent variable (Simple Linear Regression), the model is represented as:

$$y = mx + c + \epsilon$$

Where:

- y = Predicted value (dependent variable)
- x = Feature/input variable (independent variable)
- m = Slope of the line (weight or coefficient)
- c = Intercept (bias term)
- ϵ = Error term (difference between actual and predicted value)

For **multiple** independent variables (**Multiple Regression**), the equation extends to:

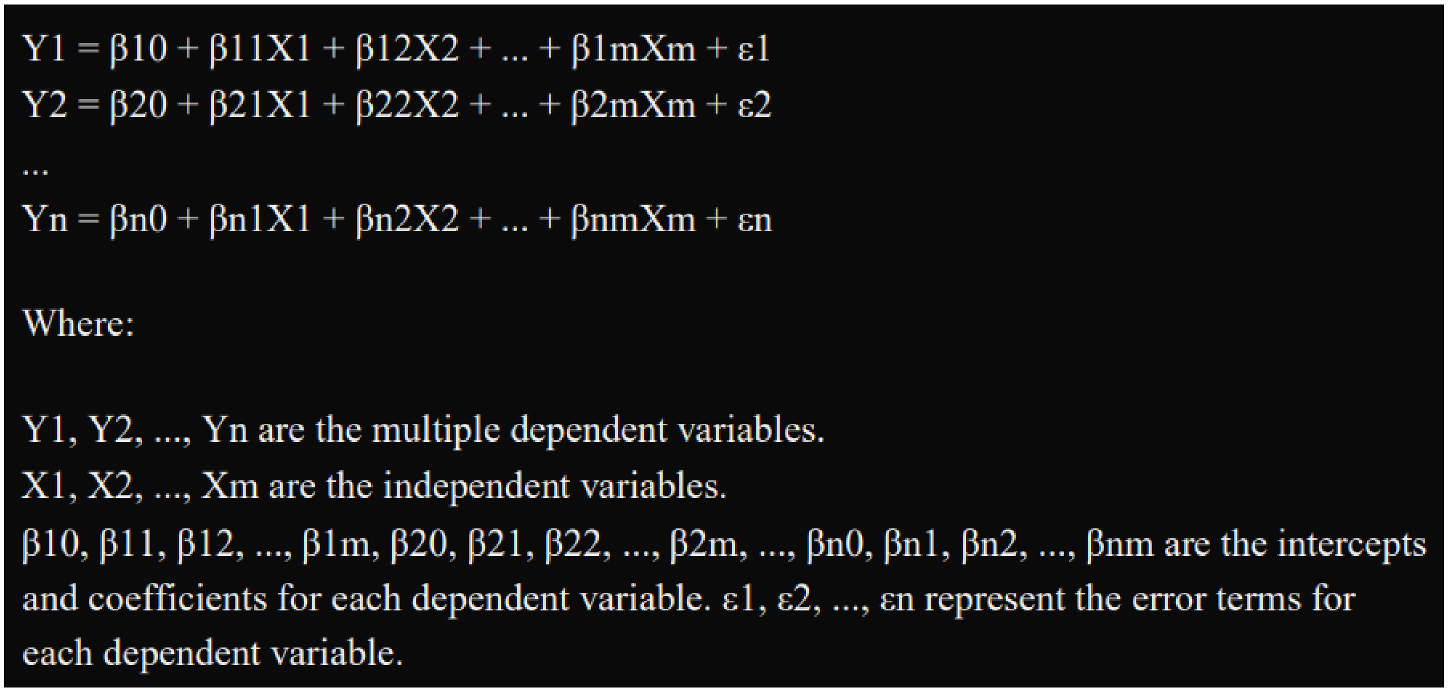


$$y = w_1x_1 + w_2x_2 + ... + w_nx_n + c + \epsilon$$

Where:

- $x_1, x_2, ...x_n$ = Independent variables (features)
- $w_1, w_2, ...w_n$ = Coefficients (weights)
- c = Intercept (bias)
- ϵ = Error term

For **multiple** dependent variables (**Multiple Outputs**), the equation extends to:



Y1 = β10 + β11X1 + β12X2 + ... + β1mXm + ε1
Y2 = β20 + β21X1 + β22X2 + ... + β2mXm + ε2
...
Yn = βn0 + βn1X1 + βn2X2 + ... + βnmXm + εn

Where:

Y1, Y2, ..., Yn are the multiple dependent variables.
X1, X2, ..., Xm are the independent variables.
β10, β11, β12, ..., β1m, β20, β21, β22, ..., β2m, ..., βn0, βn1, βn2, ..., βnm are the intercepts and coefficients for each dependent variable. ε1, ε2, ..., εn represent the error terms for each dependent variable.

2. How Linear Regression Works

Step 1: Finding the Best Fit Line

The model learns the best values for **weights (coefficients) and bias** by minimizing the error between predicted and actual values. This is typically done using:

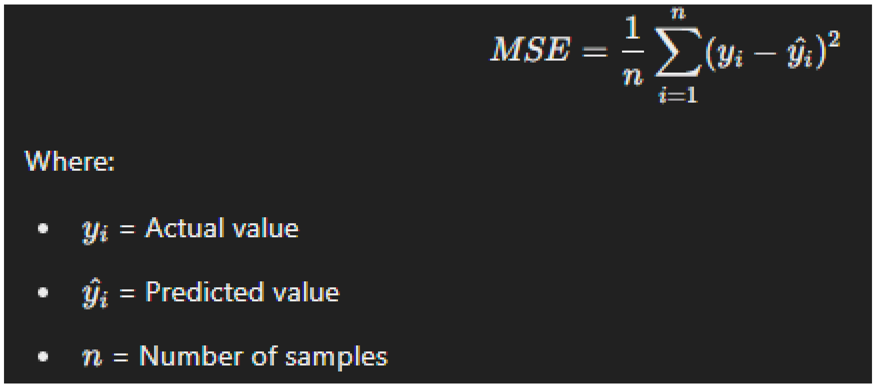
- **Ordinary Least Squares (OLS)**: Minimizes the sum of squared residuals (differences between actual and predicted values).
- **Gradient Descent**: Optimizes the model by iteratively updating weights in the direction that reduces error.

Step 2: Prediction

Once trained, the model can predict values for new input data using the learned parameters.

3. Cost Function (Loss Function)

The most commonly used loss function for linear regression is **Mean Squared Error (MSE)**:



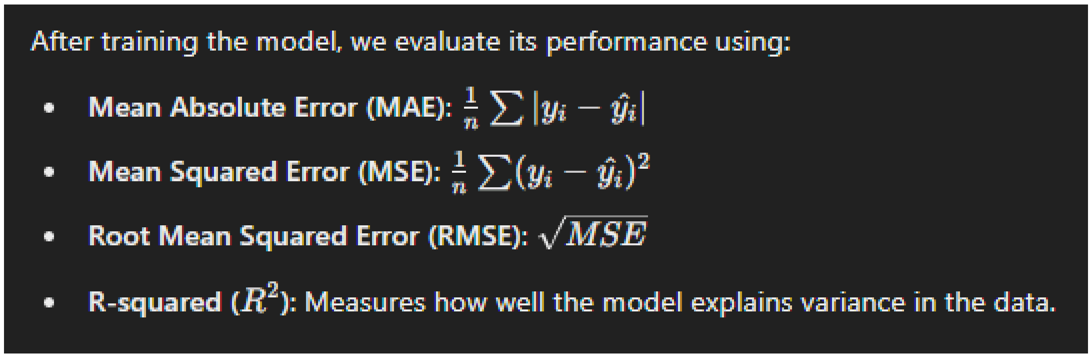
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- y_i = Actual value
- $\hat{y}_i$ = Predicted value
- n = Number of samples

The goal is to minimize this function so the predicted values are as close as possible to actual values.

4. Evaluation Metrics



After training the model, we evaluate its performance using:

- **Mean Absolute Error (MAE):** $\frac{1}{n}\sum|y_i - \hat{y}_i|$
- **Mean Squared Error (MSE):** $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$
- **Root Mean Squared Error (RMSE):** $\sqrt{MSE}$
- **R-squared (R^2):** Measures how well the model explains variance in the data.

Subset Selection

Subset selection is a technique used in regression models (especially **linear regression**) to **select a subset of relevant features (predictors)** from a larger set of features.

The goal is to improve model performance by **removing irrelevant or redundant features**, which helps in **reducing overfitting.**

Why Subset Selection?

✅ **Reduces overfitting** by keeping only the most important predictors
✅ **Enhances interpretability** by making the model simpler and easier to understand

Types of Subset Selection Methods

There are three main approaches to subset selection:

1. Best Subset Selection

- **Approach**: Evaluates all possible subsets of the predictors and selects the one that gives the best performance (e.g., lowest error).
- **Steps**:
    1. Consider all possible subsets of predictors.
    2. Fit a separate model for each subset.
    3. Compare models using evaluation metrics (e.g., AIC, BIC, Adjusted R^2).
    4. Choose the best subset.
- **Advantages**: ✅ Guarantees finding the best subset
  ✅ Works well for small datasets
- **Disadvantages**: ❌ Computationally expensive for large datasets (**if there are p features, we have to evaluate 2^p models(linear equations)**)

2. Stepwise Selection (Greedy Approach)

Stepwise selection methods iteratively **add or remove** features based on a criterion like AIC, BIC, or Adjusted R^2.

(a) Forward Stepwise Selection

- **Start with no predictors** and add one at a time.
- At each step, add the predictor that **improves the model performance the most**.
- Stop when no further improvement is possible.

✅ More computationally efficient than Best Subset Selection
❌ May miss the best subset

(b) Backward Stepwise Selection

- **Start with all predictors** and remove one at a time.
- At each step, remove the predictor that **reduces the model performance the least**.
- Stop when removing further predictors worsens the model.

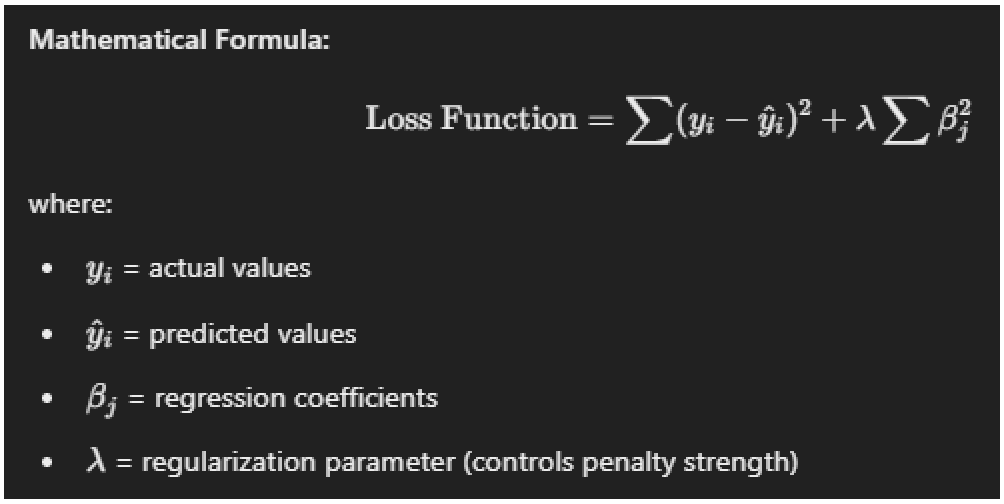
✅ Works well when p (number of features) is small
❌ Requires fitting multiple models when compared with forward stepwise subset selection method.

3. Shrinkage Methods (Regularization)

Rather than selecting a subset, these methods **shrink the coefficients** of less important features towards zero.

(a) Ridge Regression (L2 Regularization)

- Ridge Regression **adds the squared sum of coefficients** (ℓ2-norm) to the loss function.
- It **shrinks** all coefficients but **does not eliminate any** (i.e., all features remain in the model).



Mathematical Formula:

$$\text{Loss Function} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$$

where:

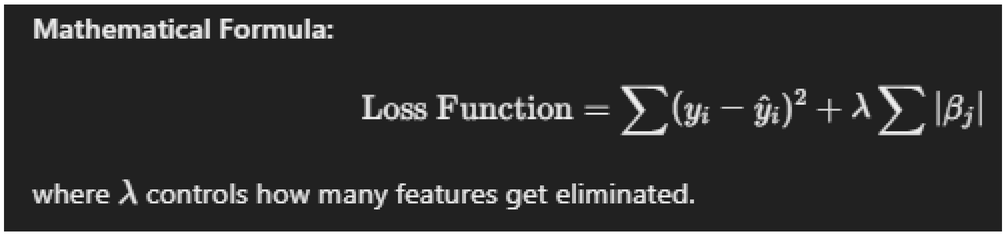
- y_i = actual values
- $\hat{y}_i$ = predicted values
- β_j = regression coefficients
- λ = regularization parameter (controls penalty strength)

Effect of Ridge Regression

- Reduces the magnitude of coefficients.
- Helps handle **multicollinearity** (correlated features).
- Does **not perform feature selection** (all features are retained).

(b) Lasso Regression (L1 Regularization)

- Lasso Regression **adds the absolute sum of coefficients** (ℓ1-norm) to the loss function.
- It **shrinks some coefficients to exactly zero**, effectively **selecting important features** and removing irrelevant ones.



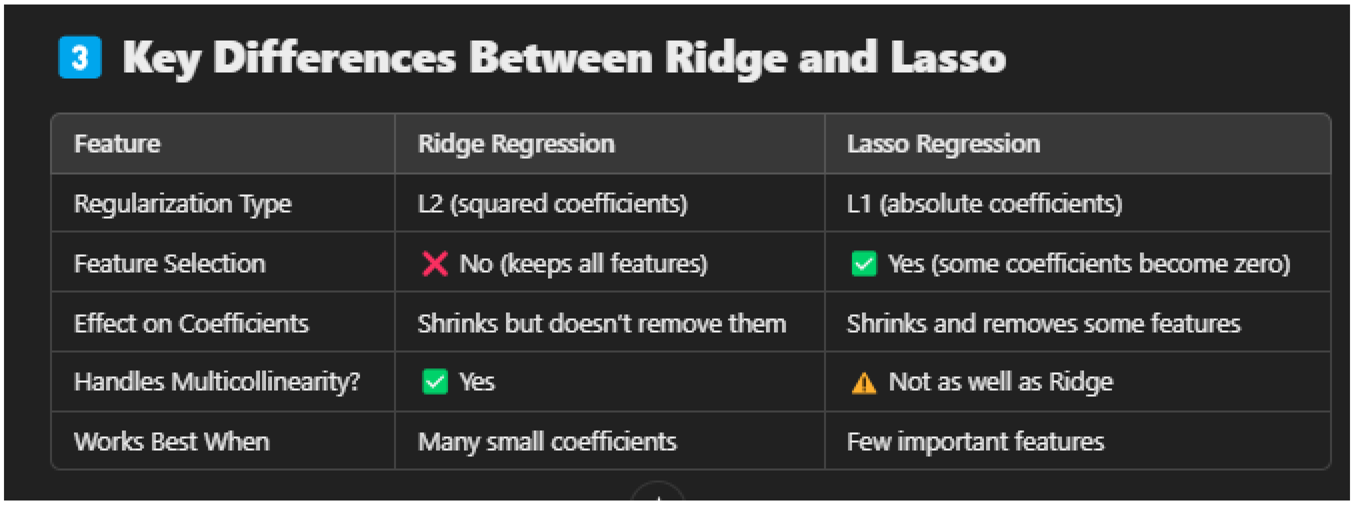
Mathematical Formula:

$$\text{Loss Function} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$$

where λ controls how many features get eliminated.

Effect of Lasso Regression

- Performs feature selection (some coefficients become zero).
- Useful when we suspect that only a few features are important.
- Works well when p (number of features) is large.

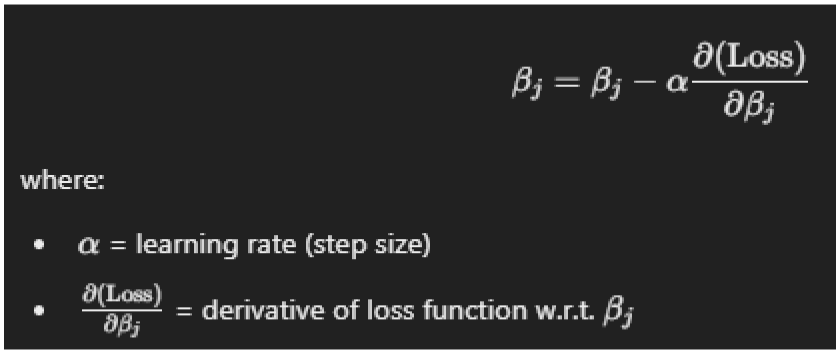


3 Key Differences Between Ridge and Lasso

Feature	Ridge Regression	Lasso Regression
Regularization Type	L2 (squared coefficients)	L1 (absolute coefficients)
Feature Selection	❌ No (keeps all features)	✅ Yes (some coefficients become zero)
Effect on Coefficients	Shrinks but doesn't remove them	Shrinks and removes some features
Handles Multicollinearity?	✅ Yes	⚠️ Not as well as Ridge
Works Best When	Many small coefficients	Few important features

What Happens After Calculating the Loss Function in Ridge and Lasso Regression?

1. **Compute the Loss**
   - The loss function (like MSE + regularization term in Ridge/Lasso) tells how far our model's predictions are from actual values.
2. **Optimize the Coefficients (β)**
   - We use **Gradient Descent** (or a similar optimization algorithm) to **minimize the loss function**.
   - The optimizer updates the coefficients βj **iteratively** to **reduce the loss**.



$$\beta_j = \beta_j - \alpha \frac{\partial(\text{Loss})}{\partial \beta_j}$$

where:

- α = learning rate (step size)
- $\frac{\partial(\text{Loss})}{\partial \beta_j}$ = derivative of loss function w.r.t. β_j

3. **Find the Best Model**
   - The final coefficients are chosen when the loss is **minimized**.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a **supervised machine learning algorithm** used for **dimensionality reduction** and **classification**.

It projects high-dimensional data onto a lower-dimensional space while maximizing class separability.

Why Use LDA?

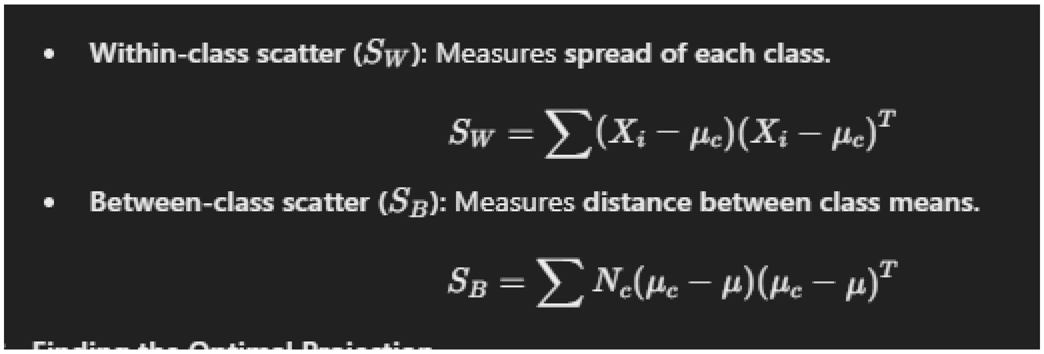
- **When you have labeled data** and want to improve classification performance.

- **Reduces dimensionality** while keeping maximum class discrimination.
- **Works well when features are correlated**, unlike Principal Component Analysis (PCA), which maximizes variance without considering class labels.
- For pattern recognition and machine learning for tasks like face recognition, image classification, and data compression.

How LDA Works

LDA finds a **new axis** that maximizes the separation between classes. It does this by:

1. **Computing the Mean for Each Class**
   - Compute the **mean vector** for each class.
2. **Computing Scatter Matrices**



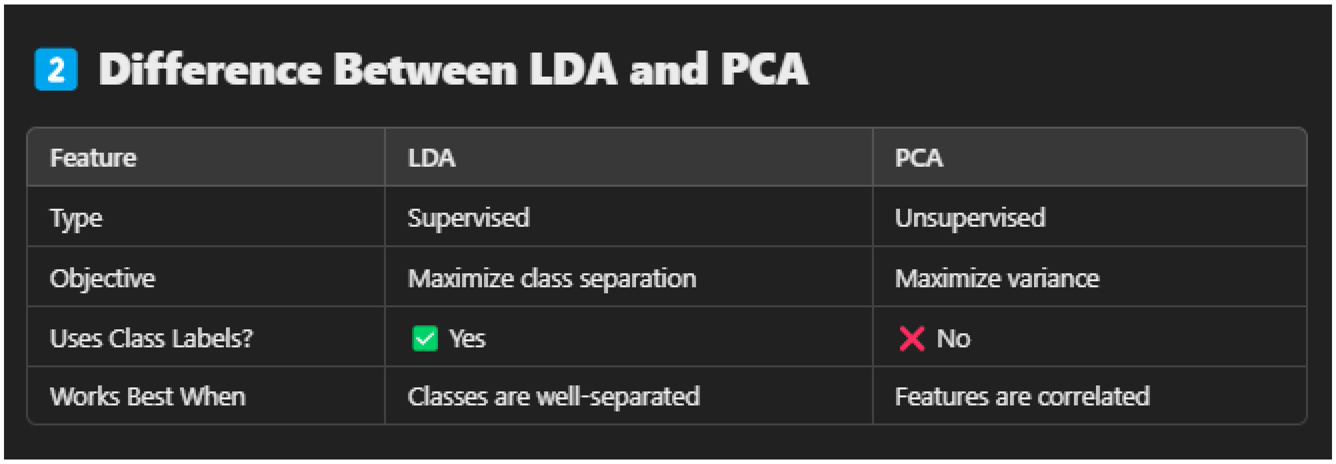
- **Within-class scatter** (S_W): Measures **spread of each class.**

$$S_W = \sum (X_i - \mu_c)(X_i - \mu_c)^T$$

- **Between-class scatter** (S_B): Measures **distance between class means.**

$$S_B = \sum N_c(\mu_c - \mu)(\mu_c - \mu)^T$$

3. **Finding the Optimal Projection**
   - Solve the **eigenvalue problem** for $S_W^{-1} S_B$.
   - Select eigenvectors corresponding to the **largest eigenvalues**.
   - These eigenvectors form the **LDA transformation matrix**.
4. **Project Data onto the New Space**
   - New feature space = original data **transformed using LDA matrix**.



2 Difference Between LDA and PCA

Feature	LDA	PCA
Type	Supervised	Unsupervised
Objective	Maximize class separation	Maximize variance
Uses Class Labels?	✅ Yes	❌ No
Works Best When	Classes are well-separated	Features are correlated

Logistic Regression

Logistic Regression is a **supervised learning algorithm** used for **Binary Classification Tasks**.

It predicts the **probability** that an instance belongs to one of the two possible classes.

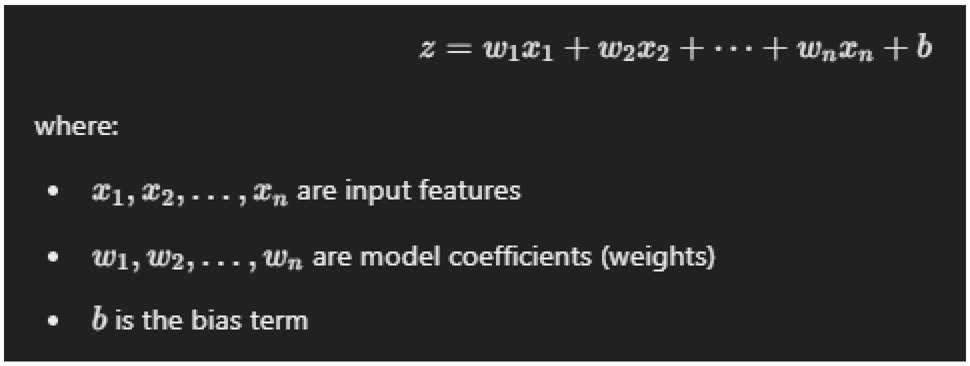
Despite its name, logistic regression is a classification algorithm, not a regression algorithm. Because Logistic Regression **models a probability** (continuous value), but the final prediction is a class label.

It uses a **sigmoid function** to map any real number to a value between **0 and 1**.

How Logistic Regression Works

🔹 Step 1: Compute Linear Combination of Features

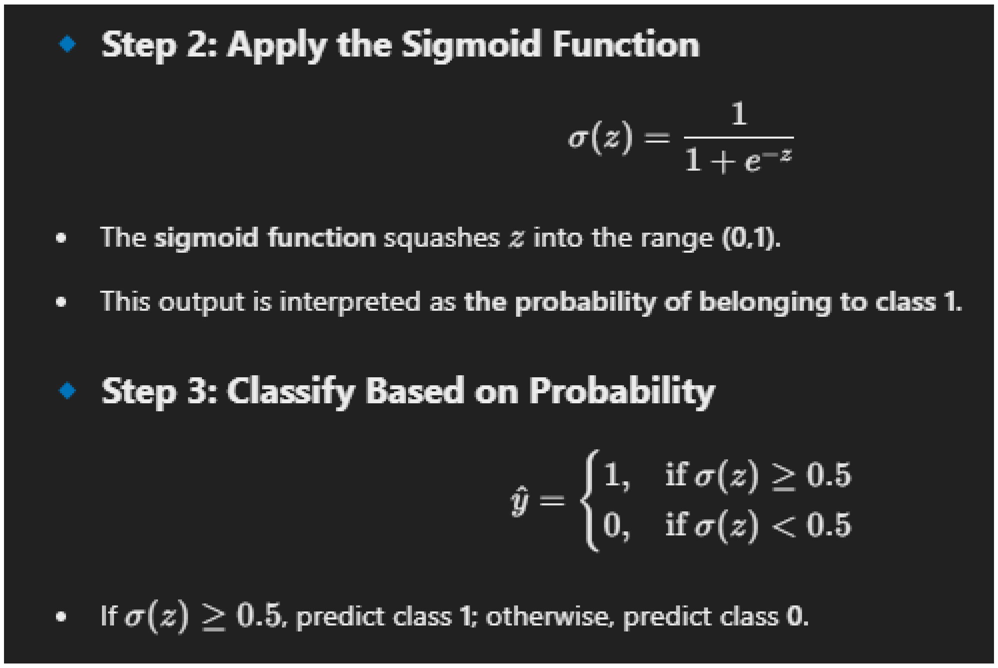
Just like Linear Regression:



$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

where:

- $x_1, x_2, \ldots, x_n$ are input features
- $w_1, w_2, \ldots, w_n$ are model coefficients (weights)
- b is the bias term



🔹 Step 2: Apply the Sigmoid Function

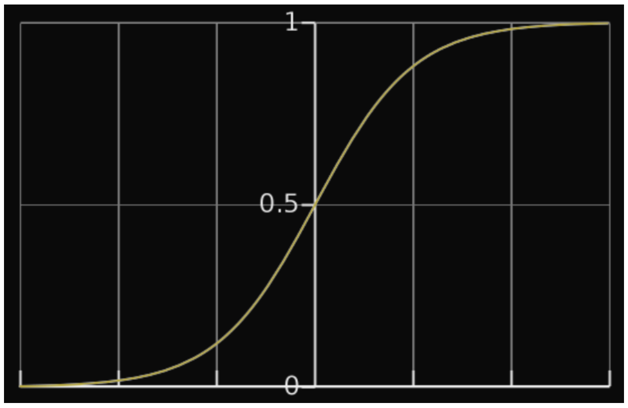
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- The **sigmoid function** squashes z into the range **(0,1)**.
- This output is interpreted as **the probability of belonging to class 1.**

🔹 Step 3: Classify Based on Probability

$$\hat{y} = \begin{cases} 1, & \text{if } \sigma(z) \geq 0.5 \\ 0, & \text{if } \sigma(z) < 0.5 \end{cases}$$

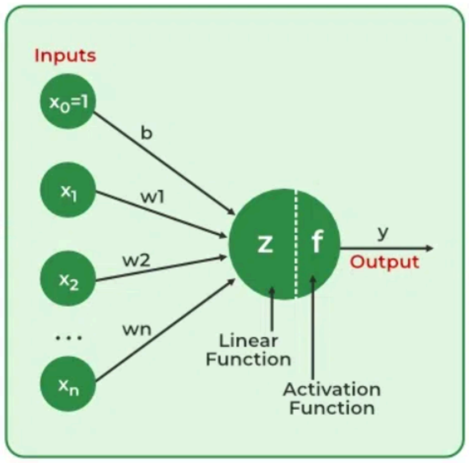
- If $\sigma(z) \geq 0.5$, predict class **1**; otherwise, predict class **0**.



(Sigmoid Function Curve)

Perceptron Learning:

- A human brain is made of billions of neurons which are the basic building blocks of the neural network.
- The decision making in the human brain is possible because of the neural network.
- Each neuron takes some form of input through an electrical signal and produces an output, which in turn is the input to another neuron and so on.
- In the ANN, the artificial neuron (a unit of ANN) is termed a **perceptron.**



- The idea of a perceptron is to use different weights to represent the importance of each input, and that the sum of the values should be greater than a threshold value before making a decision like yes or no (true or false) (0 or 1).

Structure of a Perceptron

A perceptron consists of:

- **Inputs:** Features of the data. (weights and the bias)
- **Weights:** Each input has an associated weight that determines the importance of that input.
- **Bias:** A constant added to the weighted sum of inputs to adjust the decision boundary(the line or plane which separates classes). This is the value which is altered in the back propagation process to train the ANN model, iteratively. Initially it is set to (0) or (1), and changed iteratively based on the difference b/w expected and actual output.

- **Linear Function:** A mathematical function that produces a straight-line graph and is used to compute weighted sums of inputs before applying an activation function. It is given by:

$$z = \sum_i w_i x_i + b$$

  Where:

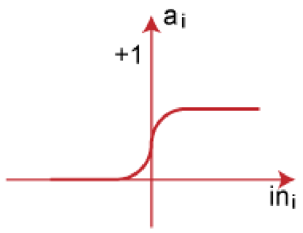
    - **wi** are the weights.
    - **xi** are the input values.
    - **b** is the bias term.
    - **z** is the weighted sum (also known as the pre-activation value).

- **Activation Function:** A function applied to the weighted sum to produce the output. It introduces non-linearity into the neural network, allowing it to model complex relationships and patterns that cannot be captured by linear functions alone. After computing the weighted sum (linear function), the activation function transforms this value to produce the output of the neuron.

  Common Activation Functions are:

  1. **Sigmoid Function:**
     - **Formula:**

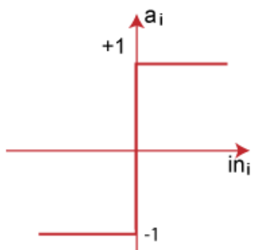
$$f(z) = \frac{1}{1+e^{-z}}$$



Sigmoid Function

     - **Range:** (0, 1)
     - **Characteristics:** Maps any real-valued number to a value between 0 and 1, making it useful for binary classification problems.
     - **Pros:** Smooth gradient, which helps in training.
     - **Cons:** Can cause vanishing gradient issues during training (gradients become very small).
  2. **ReLU (Rectified Linear Unit):**
     - **Formula:**

$$f(z) = \max(0, z)$$



Sign Function

     - **Range:** [0, ∞)
     - **Characteristics:** Outputs zero for negative inputs and passes positive inputs unchanged. It's widely used in hidden layers of deep networks.
     - **Pros:** Reduces the likelihood of vanishing gradients, computationally efficient.
     - **Cons:** Can lead to dead neurons (neurons that always output zero) if not managed properly.

3. **Tanh (Hyperbolic Tangent):**
   - **Formula:**

     $$f(z) = \tanh(z)$$

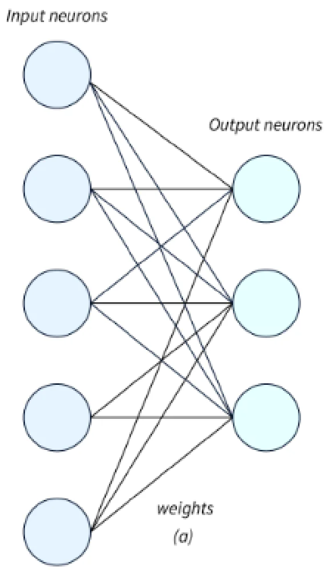
   - **Range:** (-1, 1)
   - **Characteristics:** Maps input values to a range between -1 and 1, centering the data and helping with training stability.
   - **Pros:** Centered around zero, which can make optimization easier.
   - **Cons:** Can suffer from vanishing gradient problems.

- So, the output of a single perceptron is nothing but the output of the activation function, which takes the weighted-sum(z) and the input.

<u>Types of Perceptrons:</u>

Perceptrons come in various forms and configurations, each suited for different types of tasks and complexities.

1. Single-Layer Perceptron



Description:

- The single-layer perceptron consists of one layer of neurons, also known as the input layer. It directly connects inputs to outputs with no hidden layers.
- It is capable of **solving linearly separable problems**, where classes can be separated by a straight line or hyperplane.

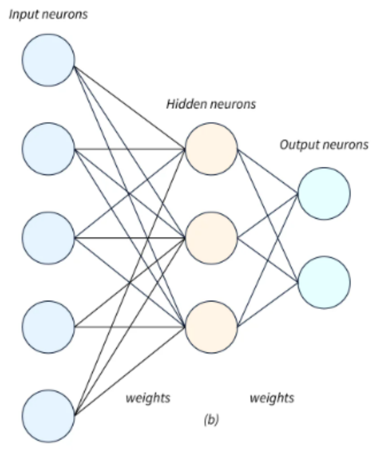
Example Use Case:

- Binary classification problems, such as distinguishing between two classes of data points that are linearly separable
  (e.g., classifying points as above or below a line).

Limitations:

- Cannot solve problems that are not linearly separable (e.g., the XOR problem).

2. Multi-Layer Perceptron (MLP)



Description:

- MLPs, also known as feedforward neural networks, consist of one or more hidden layers between the input and output layers.
- Each layer in an MLP is fully connected to the next layer, meaning each neuron in one layer is connected to every neuron in the next layer.
- It is used in **solving the non-linearly separable problems.**

Example Use Case:

- Complex classification and regression problems where data is not linearly separable. MLPs can model non-linear relationships due to the multiple layers and non-linear activation functions.

Architecture:

- **Input Layer:** Receives the input features.
- **Hidden Layers:** One or more layers where neurons perform non-linear transformations.
- **Output Layer:** Produces the final predictions or classifications.