Feature Normalization

Feature normalization is a preprocessing technique in machine learning that transforms features to a similar scale, often within a specific range. This process is essential for algorithms sensitive to feature scales, like distance-based methods (e.g., K-Nearest Neighbors and Support Vector Machines) and optimization-based methods (e.g., gradient descent in neural networks).

Why Feature Normalization is Important

1. **Improves Model Performance**: Different scales across features can lead to slower training, incorrect weights, and inaccurate results.
2. **Speeds Up Convergence**: Algorithms converge faster when data is normalized, as features contribute proportionally to the objective function.
3. **Enhances Interpretability**: Normalization makes it easier to interpret the influence of each feature on the model output.

Common Methods of Feature Normalization

1. **Min-Max Normalization**
    - Scales features to a specific range, usually between 0 and 1, or -1 and 1.
    - **Formula**:

    $$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

    - **Use Case**: Suitable for data where features have known minimum and maximum values, or when maintaining relative feature relationships is essential.
2. **Z-Score Normalization (Standardization)**
    - Transforms features to have a mean of 0 and a standard deviation of 1.
    - **Formula**:

    $$x_{\text{norm}} = \frac{x - \mu}{\sigma}$$

    - **Use Case**: Preferred when features have different units or scales, and in algorithms assuming normally distributed data (e.g., linear regression, logistic regression).
3. **Robust Normalization**

- Normalizes features based on their median and interquartile range (IQR), making it less sensitive to outliers.
- **Formula**:

$$x_{\text{norm}} = \frac{x - \text{median}}{\text{IQR}}$$

- **Use Case**: Ideal for datasets with many outliers or skewed distributions.

Example of Feature Normalization

Suppose we have a dataset with two features, **Age** and **Income**, where income has values in the thousands while age has values between 18 and 70. Applying Min-Max normalization (0-1 range) would make them comparable:

Original data:

- Age: [20, 50, 70]
- Income: [30000, 80000, 150000]

After Min-Max Normalization:

- Age: [0.05, 0.64, 1.0]
- Income: [0.0, 0.4, 1.0]

Mean of a Data Matrix

In a data matrix, the **mean** is calculated either for each **feature (column)** or each **observation (row)**, depending on the context. The mean provides a measure of the central tendency and is useful in many machine learning tasks, including normalization and standardization.

Let’s break down how to compute the mean in different contexts:

---

1. Column-wise Mean (Mean of Each Feature)

The **column-wise mean** calculates the mean of each feature (or attribute) across all observations in the dataset. This is commonly used to understand the average value of each feature.