📌 Normalization Techniques in Machine Learning

✅ What is Normalization?

Normalization is the process of **scaling numerical features** in a dataset to a **common scale** without distorting differences in the range of values.
It is crucial when features have **different units or scales**, especially for algorithms like:

- KNN
- K-Means
- SVM
- Neural Networks

🧠 Why Normalize?

- To **remove bias** from features with large magnitudes.
- To **speed up convergence** during gradient descent.
- To make distance-based algorithms (like KNN or K-Means) more effective.
- To handle features with **different units** (e.g., age vs. salary).

📊 1. Min-Max Normalization

✅ **Formula:**

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

✅ **Scales the data to a fixed range: [0, 1] or [-1, 1]**

✅ **Example:**

If X = [10, 20, 30] then:

- $X_{\min} = 10$, $X_{\max} = 30$
- $X' = \frac{X - 10}{30 - 10} = \frac{X - 10}{20}$

$X' = [0.0, 0.5, 1.0]$

✅ **Pros:**

- Preserves the shape of the original distribution.
- Works well when min and max are known and data has **no outliers**.

❌ **Cons:**

- **Sensitive to outliers**. An extreme value can stretch the range and compress other values.

📊 2. Mean Normalization

✅ **Formula:**

$$X' = \frac{X - \text{mean}(X)}{X_{\max} - X_{\min}}$$

✅ **Scales data around 0, range could still be [−1, 1] but centered at 0.**

✅ **Example:**

If X = [10, 20, 30], mean = 20, max = 30, min = 10

$$X' = \frac{X - 20}{20} = [-0.5, 0, 0.5]$$

✅ **Pros:**

- Centered around 0.
- Better than Min-Max if you want data to be **zero-centered**.

❌ **Cons:**

- Still **sensitive to outliers**.

📊 3. Max-Abs Scaling

✅ **Formula:**

$$X' = \frac{X}{|X_{\max}|}$$

✅ **Scales data in the range [-1, 1] without shifting/centering the data.**

✅ **Example:**

If X = [-10, 0, 5, 10], then:

$$X' = \frac{X}{10} = [-1.0, 0.0, 0.5, 1.0]$$

✅ **Pros:**

- **Fast** and does **not shift the data** (preserves sparsity).
- Useful for **sparse data** (e.g., text data in NLP).

❌ **Cons:**

- **Outliers affect the scaling** (like MinMax).

📊 4. Robust Scaler (Robust Normalization)

✅ **Formula:**

X′=X−median(X)IQRwhere IQR = Q3 - Q1X' = \frac{X - \text{median}(X)}{\text{IQR}} \quad \text{where IQR = Q3 - Q1}

✅ **Uses median and interquartile range (IQR) instead of mean and std.**

✅ **Example:**

If X = [1, 2, 3, 4, 100]

- Median = 3, IQR = 4 - 2 = 2

X′=X−32=[−1,−0.5,0,0.5,48.5]X' = \frac{X - 3}{2} = [-1, -0.5, 0, 0.5, 48.5]

✅ **Pros:**

- **Best choice if data contains outliers**.
- Not affected by extreme values.

❌ **Cons:**

- Doesn't guarantee values in [0,1] or [-1,1].

📈 Comparison Table:

Scaler	Outlier Sensitive	Range	Centers at 0	Use Case
MinMax	✅ Yes	[0,1]	❌ No	Features with known min/max
Mean Normalization	✅ Yes	~[-1,1]	✅ Yes	When centering is important
MaxAbs	✅ Yes	[-1,1]	❌ No	Sparse data (text data, TF-IDF)
Robust Scaler	❌ No	No fixed range	✅ Yes	Data with outliers

⚙️ When to Use What?

Situation	Scaler to Use
Data with **no outliers**	Min-Max, MaxAbs
Data with **outliers**	Robust Scaler
Data needs to be **zero-centered**	Mean Norm, Robust
Data is **sparse (mostly zeros)**	MaxAbs Scaler

📚 In Sklearn (Python Code)

```
from sklearn.preprocessing import MinMaxScaler, MaxAbsScaler, RobustScaler


# MinMax Scaler
minmax = MinMaxScaler()
X_minmax = minmax.fit_transform(X)


# MaxAbs Scaler
maxabs = MaxAbsScaler()
X_maxabs = maxabs.fit_transform(X)


# Robust Scaler
robust = RobustScaler()
X_robust = robust.fit_transform(X)
```

📌 Final Note

Normalization helps ensure that:

- All features contribute **equally**.
- Algorithms perform **efficiently and accurately**.
- Scaling should be **fit only on training data**, and **applied to test data** using the same scaler.
