Data normalization

What is Data Normalization?

Data normalization means transforming your numerical features so that they’re on the same scale.

Imagine two features:

Height (cm): [150, 160, 170, 180]

Salary ($): [30000, 50000, 70000, 150000]

Without normalization, many ML algorithms will think salary is more important just because it's bigger in value — even if height matters more.

✅After normalization, all features have equal importance for the model.

✅ Min-Max Scaling

 Result:

Scales values between 0 and 1

 Sensitive to outliers: a single large value can distort the scale.

When to use:

Data is bounded and no extreme outliers

Required for neural networks, image pixels (0–255)

Min-Max Scaling

scaler = MinMaxScaler()

df['Income_MinMax'] = scaler.fit_transform(df[['Income']])

print(df)

	Income	Income_MinMax
0	20000	0.00
1	30000	0.13
2	40000	0.25
3	50000	0.38
4	100000	1.00

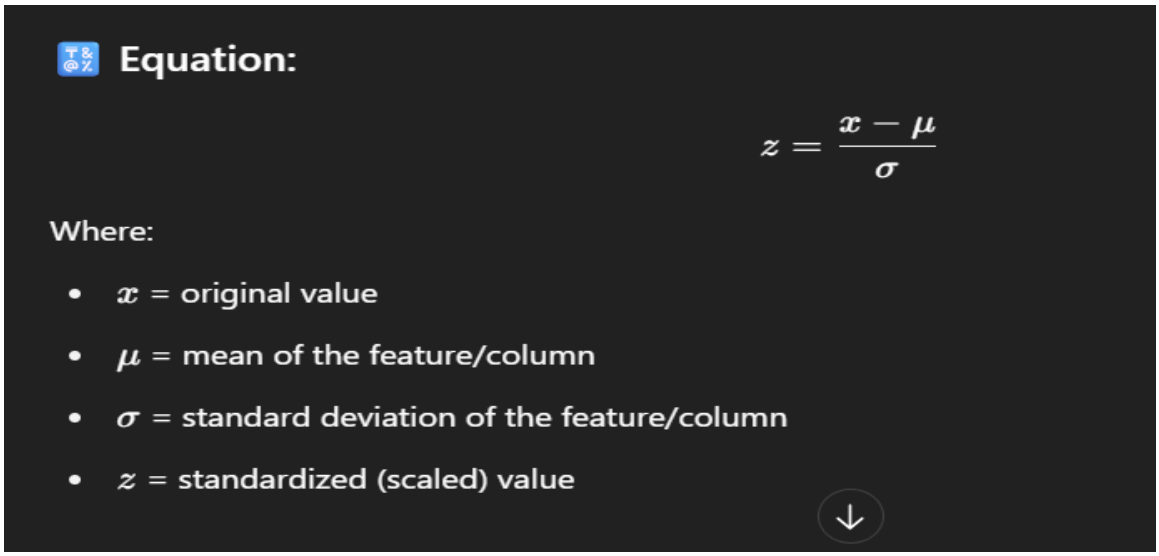
✅Standard Scaling (Z-score Normalization)

There is no fixed minimum or maximum in StandardScaler.

It centers the data around 0 with a standard deviation of 1.

data = [20, 30, 50, 80, 300]

output=[-0.83, -0.62, -0.21, 0.17, 1.48]



standard deviation

Standard deviation tells us how spread out the values in a dataset are from the mean (average).:

- If all values are very close to the mean, std is small
- If values are widely spread, std is large
- If all values are equal, std = 0

This means:

Values < mean → negative

Values > mean → positive

🔍 Result:

Mean = 0, Standard Deviation = 1

Keeps outliers but reduces impact

✅ When to use:

Data contains outliers

Used for PCA, SVM, K-Means, Linear Models

Summary: Why Std = 1?

Reason	Benefit
Equal feature weight	No one feature dominates others
Speed & stability	Improves gradient descent
Unit-free comparison	Z-scores make data universal

Code:

```
scaler_std = StandardScaler()

df['Income_Standard'] = scaler_std.fit_transform(df[['Income']])

print(df)
```