Data Science

By:
Dr. Shikha Deep

Duplicates and Scaling & Normalization

Contents :

1. Duplicates
2. Data Scaling and Normalization

What are Duplicates?

Duplicate records are rows in a dataset that appear more than once.

- They may occur due to:
  - Human error (data entry repeated)
  - System error (multiple submissions)
  - Merging datasets incorrectly

Special Cases:

1. Exact Duplicates

Every column value is identical (row is fully repeated).
Easy to detect & remove with drop_duplicates().

Example:

ID	Name	Marks	
101	Amit	80	
101	Amit	80	← duplicate row

2. Same ID, Different Values (Partial Duplicate)

Unique identifier (like **ID**) is the same, but other columns differ.

Cannot just drop blindly.

Need to decide:

- Keep the **latest record** (using timestamp).
- Take the **average/maximum/minimum**.
- Or verify from source.

Example:

ID	Name	Maths	
101	Amit	80	
101	Amit	85	← which one is correct?

3. Same Name, Different IDs (Potential Data Entry Error)

The name is the same but ID is different (maybe typo or new entry).

Need manual check OR fuzzy matching.

Example:

ID	Name	Marks	
101	Amit	80	
201	Amit	80	← two IDs, same person?

4. Nearly Duplicates (Spelling / Formatting Issues)

Values look the same but differ slightly (typos, case sensitivity, extra spaces).
Handle using **string cleaning** (str.strip(), str.lower()))

Example:

ID	Name	Marks	
101	Neha	90	
101	neha	90	← differs only in case

Scaling and Normalization

The datasets have different features often have **different ranges**.

Example:

Feature	Range
Age	18 – 60
Salary	20,000 – 1,00,000

Algorithms like KNN, Logistic Regression, Neural Networks are distance-based and are biased towards features with larger values (salary will dominate age).

Methods of Scaling & Normalization

1. Standardization (Z-score scaling)
2. Min-Max Normalization
3. Robust Scaling

1. Standardization (Z-score scaling)

Formula:

$$z = \frac{X - \mu}{\sigma}$$

- Mean = 0, Standard deviation = 1
- Keeps outliers but centers data
- Used in **regression, PCA, clustering**

CODE 1:

```
import pandas as pd

data = {
    "Age": [18, 22, 30, 45, 60],
    "Salary": [20000, 35000, 50000, 80000, 100000]
}
df = pd.DataFrame(data)
print("Original Data:\n", df)
```

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

scaler = StandardScaler()
df_standard = scaler.fit_transform(df)

scaled_df = pd.DataFrame({
    "Age": df["Age"],
    "Salary": df["Salary"],
    "Age_Standard": df_standard[:,0],
    "Salary_Standard": df_standard[:,1],
    "})
print(scaled_df)
```

OUTPUT:

Age	Salary	Age_Standard	Salary_Standard
18	20000	-1.26	-1.31
22	35000	-1.01	-0.65
30	50000	-0.52	0.00
45	80000	0.26	0.98
60	100000	1.52	0.98

How it works

Suppose column **Age = [18, 22, 30, 45, 60]**

Compute mean:

$$\mu = \frac{18 + 22 + 30 + 45 + 60}{5} = 35$$

Compute standard deviation ($\sigma \approx 16.4$).

Apply formula:

$$z(18) = \frac{18 - 35}{16.4} = -1.04$$

$$z(22) = \frac{22 - 35}{16.4} = -0.79$$

$$z(60) = \frac{60 - 35}{16.4} = 1.52$$

Now values are not "ages" anymore, they are **standardized**

Manual Code

```
import numpy as np

# Data
age = np.array([18, 22, 30, 45, 60])

# Step 1: Mean & Std
mean = np.mean(age)
std = np.std(age)

# Step 2: Standardize
z_scores = (age - mean) / std

print("Original Age:", age)
print("Standardized Age:", z_scores)
```

2. Min-Max Normalization

Normalization (also called Min–Max Scaling) transforms data into a fixed range, usually [0,1].

Formula:

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

- Scales values between **0 and 1**
- Used in **Neural Networks, KNN**

where:

X = original value

X_{min} = minimum value in the column

X_{max} = maximum value in the column

X′ = normalized value

After normalization:

Minimum value → 0

Maximum value → 1

All other values between 0 and 1

CODE:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Example dataset
data = {"Age": [18, 22, 30, 45, 60],
        "Salary": [20000, 35000, 50000, 80000, 100000]}
df = pd.DataFrame(data)
print("Original Data:\n", df)
```

```
# Normalization
scaler = MinMaxScaler()
df_normalized = scaler.fit_transform(df)

# Convert back to DataFrame
df_normalized = pd.DataFrame(df_normalized,
columns=df.columns)
print("\nAfter Normalization:\n", df_normalized)
```

How it works

Xmin=18, Xmax=60

Now formula:

For 18:

$$X' = \frac{18 - 18}{60 - 18} = 0$$

For 22:

$$X' = \frac{22 - 18}{42} = \frac{4}{42} \approx 0.095$$

For 30:

$$X' = \frac{30 - 18}{42} = \frac{12}{42} \approx 0.286$$

- For 45:

$$X' = \frac{45 - 18}{42} = \frac{27}{42} \approx 0.643$$

For 60:

$$X' = \frac{60 - 18}{42} = 1$$

New Normalized Age = [0, 0.095, 0.286, 0.643, 1]

Note: Same can solve for Salary

Person	Age (Original)	Age (Standardized)	Age (Normalized)	Salary (Original)	Salary (Standardized)	Salary (Normalized)
A	18	-1.04	0.000	20000	-1.27	0.000
B	22	-0.79	0.095	35000	-0.75	0.188
C	30	-0.31	0.286	50000	-0.24	0.375
D	45	0.61	0.643	80000	0.79	0.750
E	60	1.53	1.000	100000	1.47	1.000

3. Robust Scaling

Robust scaling is used when the dataset has outliers. Instead of using mean & standard deviation (like Standardization), it uses median & IQR (Interquartile Range).

Formula:

$$X' = \frac{X - Median}{IQR}$$

X = original value

$IQR = Q_3 - Q_1$

Ex:

Age = [18, 22, 30, 45, 60, 120]   # 120 is an outlier

where

Median = 30

Q1 (25th percentile) = 22

Q3 (75th percentile) = 60

IQR = 60 - 22 = 38

CODE:

```
import pandas as pd
from sklearn.preprocessing import RobustScaler

# Example dataset with an outlier
data = {"Age": [18, 22, 30, 45, 60, 120]}
df = pd.DataFrame(data)
print("Original Data:\n", df)
```

```
# Robust Scaling
scaler = RobustScaler()
df_robust = scaler.fit_transform(df)

# Convert back to DataFrame
df_robust = pd.DataFrame(df_robust, columns=df.columns)
print("\nAfter Robust Scaling:\n", df_robust)
```

How it Works :

For 18:

$$X' = \frac{18 - 30}{38} = -\frac{12}{38} \approx -0.32$$

For 22:

$$X' = \frac{22 - 30}{38} = -\frac{8}{38} \approx -0.21$$

For 30:

$$X' = \frac{30 - 30}{38} = 0$$

- For 45:

$$X' = \frac{45 - 30}{38} = \frac{15}{38} \approx 0.39$$

For 60:

$$X' = \frac{60 - 30}{38} = \frac{30}{38} \approx 0.79$$

For 120 (outlier):

$$X' = \frac{120 - 30}{38} = \frac{90}{38} \approx 2.37$$

OUTPUT:

	Age
0	-0.604651
1	-0.480620
2	-0.232558
3	0.232558
4	0.697674
5	2.558140

Central Limit Theorem

CLT (Central Limit Theorem)

The Central Limit Theorem states that the sampling distribution of the sample mean approaches a normal distribution as the sample size becomes large, regardless of the distribution of the population.

- If you take many random samples from any population (even non-normal),
- Calculate their means,
- Those means will form a **normal distribution** when sample size is large enough ($n \geq 30$).


Thanks!