Scaling and Standardizing in Univariate Analysis

Scaling and standardizing are preprocessing steps used in data analysis and machine learning to normalize features in a dataset. These techniques ensure that variables are on a comparable scale, which is critical for statistical models and algorithms sensitive to magnitude differences.

Key Concepts

1. Scaling

- Rescales the data to fit within a specific range, typically [0, 1] or [-1, 1].
- Ensures the range of the data values is consistent across variables.
- **Methods**:
  - Min-Max Scaling: Rescales each feature to a range of [0, 1].

2. ***Standardizing***

- Centers the data around 0 with a standard deviation of 1.
- Useful for data where the variance is important or when applying machine learning algorithms like logistic regression or k-means.
- **Formula**: Z=X−μσZ = \frac{X - \mu}{\sigma} Z=σX−μ Where:
  - XXX: Original value
  - μ\muμ: Mean of the feature
  - σ\sigmaσ: Standard deviation of the feature

Syntax for Scaling and Standardizing

- **Min-Max Scaling**:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)
```

- **Standardizing**:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data)
```

Example

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

# Sample Dataset
data = {
```

```
    "Scores": [45, 50, 67, 68, 75, 80, 85, 90, 92, 100]
}

# Create DataFrame
df = pd.DataFrame(data)

# Min-Max Scaling
min_max_scaler = MinMaxScaler()
scaled_scores = min_max_scaler.fit_transform(df[["Scores"]])

# Standardizing
standard_scaler = StandardScaler()
standardized_scores = standard_scaler.fit_transform(df[["Scores"]])

# Create new columns for scaled and standardized values
df["Scaled_Scores"] = scaled_scores
df["Standardized_Scores"] = standardized_scores

# Display Results
print("Original, Scaled, and Standardized Scores:")
print(df)
```

Output

```
Scores    Scaled_Scores    Standardized_Scores
0    45    0.000000    -1.734479
1    50    0.090909    -1.514399
2    67    0.400000    -0.676079
3    68    0.418182    -0.632065
4    75    0.545455    -0.367970
5    80    0.636364    -0.157886
6    85    0.727273     0.052199
7    90    0.818182     0.262283
8    92    0.854545     0.346311
9    100   1.000000     0.715065
```

1. **Dataset**:
    - The Scores column contains numerical values for preprocessing.
2. **Min-Max Scaling**:
    - Rescales all values to the range [0, 1] using the formula: Xscaled=X−min(X)max(X)−min(X)X_{\text{scaled}} = \frac{X - \text{min}(X)}{\text{max}(X) - \text{min}(X)}Xscaled =max(X)−min(X)X−min(X)

   - Ensures that the minimum score (45) becomes 0 and the maximum score (100) becomes 1.
3. **Standardizing**:
   - Centers the values around 0 with a standard deviation of 1.
   - Each score is transformed using the formula: Z=X−μσZ = \frac{X - \mu}{\sigma}Z=σX−μ
   - This ensures that the transformed data follows a standard normal distribution.
4. **Output**:
   - The DataFrame contains three columns:
     - Original Scores.
     - Min-Max Scaled values in Scaled_Scores.
     - Standardized values in Standardized_Scores.

Key Insights from Output

1. **Scaled Scores**:
   - Range is [0, 1].
   - Useful for algorithms requiring bounded input like neural networks.
2. **Standardized Scores**:
   - Mean is approximately 0.
   - Standard deviation is 1.
   - Useful for algorithms like Principal Component Analysis (PCA), k-means clustering, and gradient-based optimization.

Use Cases

- **Scaling**:
  - When working with data that has varying units or scales.
  - Suitable for distance-based algorithms like k-NN and SVM.

- **Standardizing**:
  - When features have different variances or means.
  - Necessary for algorithms assuming a Gaussian distribution, like logistic regression or PCA.