Scaling Techniques Documentation

Introduction

Scaling is a crucial step in data preprocessing, especially for machine learning algorithms that are sensitive to the magnitude of feature values. This document outlines the scaling techniques applied to our training and testing datasets.

StandardScaler

Description: Standardizes features by removing the mean and scaling to unit variance.

Use Cases: Preferred when the distribution of features is normal.

Code Snippet:

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

MinMaxScaler

Description: Scales features to a given range, typically [0, 1].

Use Cases: Preferred when the distribution is not normal and the dataset contains outliers.

Code Snippet:

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

Implementation

The scaling techniques were applied as follows:

1. StandardScaler was used for features with a normal distribution.

2. MinMaxScaler was applied to features with outliers or non-normal distribution.

Example Code:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler


# Instantiate the scalers

standard_scaler = StandardScaler()

minmax_scaler = MinMaxScaler()


# Apply StandardScaler

X_train_standard_scaled = standard_scaler.fit_transform(X_train)

X_test_standard_scaled = standard_scaler.transform(X_test)


# Apply MinMaxScaler

X_train_minmax_scaled = minmax_scaler.fit_transform(X_train)

X_test_minmax_scaled = minmax_scaler.transform(X_test)
```

Conclusion

The applied scaling techniques have ensured that our data is appropriately scaled, enhancing the performance and reliability of our machine learning models.