How can data be standardized in Python?

Authored by

stats writer

May 1, 2024

Data standardization in Python refers to the process of transforming data into a common format to ensure consistency and compatibility across different systems. This involves converting data into a specific structure or type, such as numerical or categorical, and normalizing data values to a standard scale. In Python, this can be achieved through various methods such as using libraries like Pandas or NumPy, which provide functions for data manipulation and transformation. Other options include creating custom functions or using built-in methods like .map() or .apply(). Standardizing data in Python is essential for data analysis and machine learning tasks, as it allows for accurate comparisons and meaningful insights to be drawn from the data.

Standardize Data in Python (With Examples)

To standardize a dataset means to scale all of the values in the dataset such that the mean value is 0 and the standard deviation is 1.

We use the following formula to standardize the values in a dataset:

xnew = (xi - x) / s

where:

xi: The ith value in the datasetx: The sample means: The sample standard deviation

We can use the following syntax to quickly standardize all of the columns of a pandas DataFrame in Python:

(df-df.mean())/df.std()

The following examples show how to use this syntax in practice.

Example 1: Standardize All Columns of DataFrame

The following code shows how to standardize all columns in a pandas DataFrame:

```
import pandas as pd

#create data frame
df = pd.DataFrame({'y': ,
'x1': ,
'x2': ,
'x3': })

#view data frame
df
```

```
y x1 x2 x3
0 8 5 11 2
1 12 7 8 2
2 15 7 10 3
3 14 9 6 2
4 19 12 6 5
5 23 9 5 5
```

6 25 9 9 7

7 29 4 12 9

#standardize the values in each column

df_new = (df-df.mean())/df.std()

#view new data frame

df_new

y x1 x2 x3

0 -1.418032 -1.078639 1.025393 -0.908151

1 -0.857822 -0.294174 -0.146485 -0.908151

2 -0.437664 -0.294174 0.634767 -0.525772

3 -0.577717 0.490290 -0.927736 -0.908151

4 0.122546 1.666987 -0.927736 0.238987

5 0.682756 0.490290 -1.318362 0.238987

6 0.962861 0.490290 0.244141 1.003746

7 1.523071 -1.470871 1.416019 1.768505

We can verify that the mean and standard deviation of each column is equal to 0 and 1, respectively:

#view mean of each column

df_new.mean()

```
y 0.000000e+00
x1 2.775558e-17
x2 -4.163336e-17
x3 5.551115e-17
dtype: float64

#view standard deviation of each column
df_new.std()

y 1.0
x1 1.0
x2 1.0
x3 1.0
dtype: float64
```

Example 2: Standardize Specific Columns of DataFrame

Sometimes you may only want to standardize specific columns in a DataFrame.

For example, for many you may only want to standardize the predictor variables before fitting a certain model to the data.

The following code shows how to standardize specific columns in a pandas DataFrame:

```
import pandas as pd

#create data frame
df = pd.DataFrame({'y': ,
'x1': ,
'x2': ,
'x3': })

#view data frame
df

y x1 x2 x3
0 8 5 11 2
1 12 7 8 2
2 15 7 10 3
3 14 9 6 2
4 19 12 6 5
5 23 9 5 5
6 25 9 9 7
7 29 4 12 9

#define predictor variable columns
df_x = df]

#standardize the values for each predictor variable
```

```
df] = (df_x-df_x.mean())/df_x.std()

#view new data frame
df

y x1 x2 x3
0 8 -1.078639 1.025393 -0.908151
1 12 -0.294174 -0.146485 -0.908151
2 15 -0.294174 0.634767 -0.525772
3 14 0.490290 -0.927736 -0.908151
4 19 1.666987 -0.927736 0.238987
5 23 0.490290 -1.318362 0.238987
6 25 0.490290 0.244141 1.003746
7 29 -1.470871 1.416019 1.768505
```

Notice that the "y" column remains unchanged, but the columns "x1", "x2", and "x3" are all standardized.

We can verify that the mean and standard deviation of each predictor variable column is equal to 0 and 1, respectively:

```
#view mean of each predictor variable column
df].mean()
```

x1 2.775558e-17

x2 -4.163336e-17

x3 5.551115e-17

dtype: float64

#view standard deviation of each predictor variable column

df].std()

x1 1.0

x2 1.0

x3 1.0

dtype: float64