How can data be normalized in Python?

Authored by

stats writer

May 5, 2024

Data normalization is a process used to standardize data in order to remove any inconsistencies and make it more organized and usable. In Python, data normalization can be achieved through various methods such as scaling, standardization, and feature engineering. Scaling involves transforming numerical data to a specific range, while standardization involves converting data to have a mean of 0 and a standard deviation of 1. Feature engineering involves creating new features or modifying existing ones to better represent the data. These methods can be implemented using various libraries and functions in Python, such as Scikit-learn and Pandas. By normalizing data in Python, it becomes more suitable for analysis and modeling, leading to more accurate and meaningful results.

Normalize Data in Python

Often in statistics and machine learning, we normalize variables such that the range of the values is between 0 and 1.

The most common reason to normalize variables is when we conduct some type of multivariate analysis (i.e. we want to understand the relationship between several predictor variables and a response variable) and we want each variable to contribute equally to the analysis.

When variables are measured at different scales, they often do not contribute equally to the analysis. For example, if the values of one variable range from 0 to 100,000 and the values of another variable range from 0 to 100, the variable with the larger range will be given a

larger weight in the analysis.

By normalizing the variables, we can be sure that each variable contributes equally to the analysis.

To normalize the values to be between 0 and 1, we can use the following formula:

xnorm = (xi - xmin) / (xmax - xmin)

where:

xnorm: The ith normalized value in the datasetxi: The ith value in the datasetxmax: The minimum value in the datasetxmin: The maximum value in the dataset

The following examples show how to normalize one or more variables in Python.

Example 1: Normalize a NumPy Array

The following code shows how to normalize all values in a NumPy array:

```
import numpy as np

#create NumPy array
```

```
data = np.array([])

#normalize all values in array
data_norm = (data - data.min())/ (data.max() - data.min())

#view normalized values
data_norm

array([])
```

Each of the values in the normalized array are now between 0 and 1.

Example 2: Normalize All Variables in Pandas DataFrame

The following code shows how to normalize all variables in a pandas DataFrame:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'points': ,
'assists': ,
'rebounds': })

#normalize values in every column
```

df_norm = (df-df.min())/ (df.max() - df.min())

#view normalized DataFrame
df_norm

points assists rebounds
0 0.764706 0.125 0.857143
1 0.000000 0.375 0.428571
2 0.176471 0.375 0.714286
3 0.117647 0.625 0.142857
4 0.411765 1.000 0.142857
5 0.647059 0.625 0.000000
6 0.764706 0.625 0.571429
7 1.000000 0.000 1.000000

Each of the values in every column are now between 0 and1.

Example 3: Normalize Specific Variables in Pandas DataFrame

The following code shows how to normalize a specific variables in a pandas DataFrame:

import pandas as pd

#create DataFrame

```
df = pd.DataFrame({'points': ,
'assists': ,
'rebounds': })

define columns to normalize
x = df.iloc

#normalize values in first two columns only
df.iloc = (x-x.min())/ (x.max() - x.min())

#view normalized DataFrame
df

points assists rebounds
0 0.764706 0.125 11
1 0.000000 0.375 8
2 0.176471 0.375 10
3 0.117647 0.625 6
4 0.411765 1.000 6
5 0.647059 0.625 5
6 0.764706 0.625 9
7 1.000000 0.000 12
```

Notice that just the values in the first two columns are normalized.

The following tutorials provide additional information on normalizing data: