How can I create a lag column in Pandas, and what are some examples of using it?

Authored by

stats writer

June 27, 2024

Creating a lag column in Pandas involves shifting the values in a specific column by a certain number of rows. This can be achieved using the .shift() method in Pandas. The resulting column will have the same length as the original column, with the first few rows containing null values.

One example of using a lag column is in time series analysis, where it can be used to calculate the difference between the current value and the previous value in a dataset. This can be useful in identifying trends and patterns in the data. Another example is in financial analysis, where a lag column can be used to calculate the change in stock prices over a certain period of time.

Overall, creating a lag column in Pandas can be a powerful tool for data analysis, allowing for the manipulation and comparison of data within a dataset. It can be applied in various industries and scenarios, making it a versatile function in data analysis.

Create a Lag Column in Pandas (With Examples)

You can use the shift() function in pandas to create a column that displays the lagged values of another column.

This function uses the following basic syntax:

df = df.shift(1)

Note that the value in the shift() function indicates the number of values to calculate the lag for.

The following example shows how to use this syntax in practice.

Example: Create a Lag Column in Pandas

Suppose we have the following pandas DataFrame that shows the sales made by some store on 10 consecutive days:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'day': ,
'sales': })
#view DataFrame
print(df)
```

```
day sales
0 1 18
1 2 10
2 3 14
3 4 13
4 5 19
5 6 24
6 7 25
7 8 29
8 9 15
9 10 18
```

We can use the shift() function to create a lag column

that displays the sales for the previous day for each row:

```
#add column that represents lag of sales column
df = df.shift(1)

#view updated DataFrame
print(df)

day sales sales_previous_day
0 1 18 NaN
1 2 10 18.0
2 3 14 10.0
3 4 13 14.0
4 5 19 13.0
5 6 24 19.0
6 7 25 24.0
7 8 29 25.0
8 9 15 29.0
9 10 18 15.0
```

Here's how to interpret the output:

The first value in the lag column is NaN since there is no prior value in the sales column.The second value in

the lag column is 18 since this is the prior value in the sales column.The third value in the lag column is 10 since this is the prior value in the sales column.

And so on.

Note that we can also add multiple lag columns to the DataFrame if we'd like:

```
#add two lag columns
df = df.shift(1)
df = df.shift(2)

#view updated DataFrame
print(df)
```

```
day sales sales_previous_day sales_previous_day2
0 1 18 NaN NaN
1 2 10 18.0 NaN
2 3 14 10.0 18.0
3 4 13 14.0 10.0
4 5 19 13.0 14.0
5 6 24 19.0 13.0
6 7 25 24.0 19.0
7 8 29 25.0 24.0
```

8 9 15 29.0 25.0

9 10 18 15.0 29.0

You can use the same general approach to add as many lag columns as you'd like.

Note: To create a lead column, simply use negative values in the shift() function.

The following tutorials explain how to perform other common tasks in pandas: