How can I use groupby in Pandas to calculate the mean while also taking into account NaN values?

Authored by

stats writer

June 24, 2024

Groupby in Pandas is a useful tool for organizing and analyzing data in a DataFrame. It allows us to group data by a certain column or set of columns and perform operations, such as calculating the mean, on the grouped data. When using groupby to calculate the mean, we can also specify how we want to handle NaN (not a number) values. By default, NaN values are excluded from the calculation, but we can also choose to include them by specifying the "dropna=False" parameter. This allows us to accurately calculate the mean while taking into account any missing or incomplete data in our dataset. Thus, groupby in Pandas gives us the flexibility to handle NaN values in our calculations, making our data analysis more accurate and comprehensive.

Pandas: Use Groupby to Calculate Mean and Not Ignore NaNs

When using the pandas groupby() function to group by one column and calculate the mean value of another column, pandas will ignore NaN values by default.

If you would instead like to display NaN if there are NaN values present in a column, you can use the following basic syntax:

df.groupby('team').agg({'points': lambda x: x.mean(skipna=False)})

This particular example will group the rows of the DataFrame by the team column and then calculate the mean value of the points column without ignoring NaN values.

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

Example: Use pandas groupby() and Don't Ignore NaNs

Suppose we have the following pandas DataFrame that contains information about various basketball players:

```
import pandas as pd
import numpy as np

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

#view DataFrame
print(df)

team points
0 A 15.0
1 A NaN
2 A 24.0
3 A 25.0
4 A 20.0
5 B 35.0
6 B 34.0
```

7 B 19.0
8 B 14.0
9 B 12.0

Suppose we use the following syntax to calculate the mean value of points, grouped by team:

#calculate mean of points, grouped by team
df.groupby('team').mean()

team
A 21.0
B 22.8
Name: points, dtype: float64

Notice that the mean value of points for each team is returned, even though there is a NaN value for team A in the points column.

By default, pandas simply ignores the NaN value when calculating the mean.

If you would instead like to display NaN as the mean value if there are indeed NaNs present, you can use the following syntax:

```
#calculate mean points value grouped by team and don't ignore NaNs
df.groupby('team').agg({'points': lambda x: x.mean(skipna=False)})
```

```
points
team
A NaN
B 22.8
```

Notice that a NaN value is returned as the mean points value for team A this time.

By using the argument skipna=False, we told pandas not to ignore the NaN values when calculating the mean.