How can I use the dropna() function in Pandas with a threshold value?

Authored by

stats writer

June 24, 2024

The dropna() function in Pandas is a powerful tool that allows users to remove missing or null values from a dataset. This function can be used with a threshold value, which specifies the minimum number of non-null values required for a row or column to be kept. This means that any rows or columns with a number of missing values above the threshold will be dropped. By setting a threshold value, users can effectively filter out incomplete or irrelevant data from their dataset, resulting in a more accurate and reliable analysis. The dropna() function with a threshold value is a valuable feature in Pandas that helps users clean and manipulate their data efficiently.

Pandas: Use dropna() with thresh

You can use the dropna() function to drops rows from a pandas DataFrame that contain missing values.

You can also use the thresh argument to specify the minimum number of non-NaN values that a row or column must have in order to be kept in the DataFrame.

Here are the most common ways to use the thresh argument in practice:

Method 1: Only Keep Rows with Minimum Number of non-NaN Values

#only keep rows with at least 2 non-NaN values
df.dropna(thresh=2)

Method 2: Only Keep Rows with Minimum % of non-NaN Values

#only keep rows with at least 70% non-NaN values
df.dropna(thresh=0.7*len(df.columns))

Method 3: Only Keep Columns with Minimum Number of non-NaN Values

#only keep columns with at least 6 non-NaN values
df.dropna(thresh=6, axis=1)

Method 4: Only Keep Columns with Minimum % of non-NaN Values

#only keep columns with at least 70% non-NaN values
df.dropna(thresh=0.7*len(df), axis=1)

The following examples show how to use each method in practice with the following pandas DataFrame:

```
import pandas as pd
import numpy as np

#create DataFrame
df = pd.DataFrame({'team': ,
'points': ,
'assists': ,
```

'rebounds': })

#view DataFrame
print(df)

team points assists rebounds
0 A 18.0 5.0 11.0
1 B NaN NaN NaN
2 C 19.0 NaN 10.0
3 D 14.0 9.0 6.0
4 E 14.0 NaN 6.0
5 F 11.0 9.0 5.0
6 G 20.0 9.0 9.0
7 H NaN 4.0 NaN

Example 1: Only Keep Rows with Minimum Number of non-NaN Values

We can use the following syntax to only keep the rows in the DataFrame that have at least 2 non-NaN values:

#only keep rows with at least 2 non-NaN values
df.dropna(thresh=2)

team points assists rebounds
0 A 18.0 5.0 11.0

2 C 19.0 NaN 10.0

3 D 14.0 9.0 6.0

4 E 14.0 NaN 6.0

5 F 11.0 9.0 5.0

6 G 20.0 9.0 9.0

7 H NaN 4.0 NaN

Notice that the row in index position 1 has been dropped since it only had 1 non-NaN value in the entire row.

Example 2: Only Keep Rows with Minimum % of non-NaN Values

We can use the following syntax to only keep the rows in the DataFrame that have at least 70% non-NaN values:

```
#only keep rows with at least 70% non-NaN values
df.dropna(thresh=0.7*len(df.columns))
```

team points assists rebounds

0 A 18.0 5.0 11.0

2 C 19.0 NaN 10.0

3 D 14.0 9.0 6.0

4 E 14.0 NaN 6.0

5 F 11.0 9.0 5.0
6 G 20.0 9.0 9.0

Notice that the rows in index positions 1 and 7 have been dropped since those rows did not have at least 70% of the values as non-NaN values.

Example 3: Only Keep Columns with Minimum Number of non-NaN Values

We can use the following syntax to only keep the columns in the DataFrame that have at least 6 non-NaN values:

```
#only keep columns with at least 6 non-NaN values
df.dropna(thresh=6, axis=1)
```

```
team points rebounds
0 A 18.0 11.0
1 B NaN NaN
2 C 19.0 10.0
3 D 14.0 6.0
4 E 14.0 6.0
5 F 11.0 5.0
6 G 20.0 9.0
```

7 H NaN NaN

Notice that the 'assists' column has been dropped because that column did not have at least 6 non-NaN values in the column.

Example 4: Only Keep Columns with Minimum % of non-NaN Values

We can use the following syntax to only keep the columns in the DataFrame that have at least 70% non-NaN values:

```
#only keep columns with at least 70% non-NaN values
df.dropna(thresh=0.7*len(df), axis=1)
```

```
  team  points  rebounds
0    A    18.0      11.0
1    B     NaN       NaN
2    C    19.0      10.0
3    D    14.0       6.0
4    E    14.0       6.0
5    F    11.0       5.0
6    G    20.0       9.0
7    H     NaN       NaN
```

Notice that the 'assists' column has been dropped because that column did not have at least 70% non-NaN values in the column.

Note: You can find the complete documentation for the pandas dropna() function .

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