Create time series dataframe with sliding windows

Asked 4 years, 2 months ago Modified 4 years, 2 months ago Viewed 952 times

2

I have a dataset that looks like this:

```
     A  B
5/8  2  3
6/8  4  2
7/8  3  5
8/8  3  2
```

and I want to finish like this

```
index1 index2   A  B
5/8       5/8   2  3
          6/8   4  2
6/8       6/8   4  2
          7/8   3  5
7/8       7/8   3  5
          8/8   3  2
 etc.
```

and also an equivalent that would take numeric indexes. This way I can decide whether flatten the data or create a 3d array for the ML training.

I have done it with df.iterrows() but it so slow. I also tried by making this code:

```
  def addDatas(x,df,window):
    global dfOo #Dataset to create
    if len(x)==window:
      y = df.loc[x.index];
      y.DateStarted  = df.loc[x.index[-1]].created #index1 in table presented
      dfOo = dfOo.append(y)
      return 0;
  dfOo= pd.DataFrame();
  #created is the date index in the first table
  dfTargets.rolling("5s",on="created").apply(lambda x :
addDatas(x,dfTargets,5))
```

Both of these solutions work but they aren´t fast enough and not usable with big chunks of data. I can help but think that there must be an easier way to do this that I don´t know.

time-series pandas

Share
Improve this question
Follow

edited Sep 29, 2020 at 22:38
desertnaut
60.2k ● 31 ● 151 ● 176

asked Sep 29, 2020 at 16:14
luso97
39 ● 2

2 Answers

Sorted by: Highest score (default)

1

The following will work on any sortable index. It does create a copy of the dataframe in memory so that is a drawback of this approach if you are memory restricted.

```
import pandas as pd

# Minimal example
df = pd.DataFrame(data={'index':['5/8','6/8','7/8','8/8'],'A':[2,4,3,3],'B':
[3,2,5,2]})

# Create a shifted version of the index 'index' column
df['index_2'] = df['index'].shift()

# Copy to df2, renaming columns and dropping null value (first shifted row)
df2 = df.copy().rename({'index':'index_2','index_2':'index'},axis=1).dropna()

# In original df overwrite index_2 to be equal to index column
df['index_2'] = df['index']

# Concatenate, set index, and sort by index
pd.concat([df,df2]).set_index(['index','index_2']).sort_index()
```

Output:

```
                A   B
index   index_2
5/8     5/8     2   3
        6/8     4   2
6/8     6/8     4   2
        7/8     3   5
7/8     7/8     3   5
        8/8     3   2
8/8     8/8     3   2
```

Share Improve this answer Follow

answered Sep 29, 2020 at 23:26
quizzical_panini
434 ● 2 ● 9

great!, that´s much simpler than what I did. thank you very much – luso97 Sep 30, 2020 at 7:07

I'd like to present a solution using `np.repeat`. We'll first load the data:

0

```
df = pd.DataFrame({'A':[2,4,3,3], 'B':[3,2,5,2]}, index=['5/8', '6/8', '7/8',
'8/8'])
```

We produce first a list, call it *xi*, that has across its length values 2, except the first and last element.

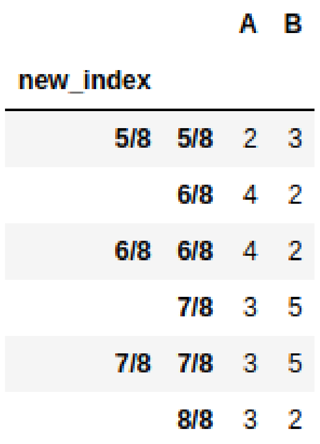
```
xi=[2]*len(df)
xi[0]=1
xi[-1]=1
```

This list will be used in `np.repeat` to repeat the desired elements. Basically, the following gives the desired data, except that an index missing:

```
ndf = df.loc[np.repeat(df.index.values, xi)]
```

The following prepares the first-level index:

```
ndf.set_index([np.repeat(ndf.index, [2,0]*int(len(ndf)/2)), ndf.index])
```



		A	B
new_index			
5/8	5/8	2	3
	6/8	4	2
6/8	6/8	4	2
	7/8	3	5
7/8	7/8	3	5
	8/8	3	2

Share

Improve this answer

Follow

edited Oct 1, 2020 at 20:50

answered Sep 30, 2020 at 13:04

Ruthger Righart

4,901 ● 2 ● 31 ● 35