top_and_bottom_performing

October 7, 2018

1 Top and Bottom Performing

Let's look at how we might get the top performing stocks for a single period. For this example, we'll look at just a single month of closing prices:

```
In [1]: import pandas as pd

        month = pd.to_datetime('02/01/2018')
        close_month = pd.DataFrame(
            {
                'A': 1,
                'B': 12,
                'C': 35,
                'D': 3,
                'E': 79,
                'F': 2,
                'G': 15,
                'H': 59},
            [month])

        close_month
```

Out[1]:

	A	B	C	D	E	F	G	H
2018-02-01	1	12	35	3	79	2	15	59

`close_month` gives use the prices for the month of February, 2018 for all the stocks in this universe (A, B, C, ...). Looking at these prices, we can see that the top 2 performing stocks for that month was E and H with the prices 79 and 59.

To get this using code, we can use the `Series.nlargest` function. This function returns the items with the n largest numbers. For the example we just talked about, our n is 2.

```
In [2]: try:
            # Attempt to run nlargest
            close_month.nlargest(2)
        except TypeError as err:
            print('Error: {}'.format(err))
```

```
Error: nlargest() missing 1 required positional argument: 'columns'
```

What happeened here? It turns out we're not calling the `Series.nlargest` function, we're actually calling `DataFrame.nlargest`, since `close_month` is a DataFrame. Let's get the Series from the dataframe using `.loc[month]`, where `month` is the 2018-02-01 index created above.

```
In [3]: close_month.loc[month].nlargest(2)

Out[3]: E    79
        H    59
        Name: 2018-02-01 00:00:00, dtype: int64
```

Perfect! That gives us the top performing tickers for that month. Now, how do we get the bottom performing tickers? There's two ways to do this. You can use Panda's `Series.nsmallest` function or just flip the sign on the prices and then apply `DataFrame.nlargest`. Either way is fine. For this course, we'll flip the sign with nlargest. This allows us to reuse any funtion created with nlargest to get the smallest.

To get the bottom 2 performing tickers from `close_month`, we'll flip the sign.

```
In [4]: (-1 * close_month).loc[month].nlargest(2)

Out[4]: A   -1
        F   -2
        Name: 2018-02-01 00:00:00, dtype: int64
```

That gives us the bottom performing tickers, but not the actual prices. To get this, we can flip the sign from the output of nlargest.

```
In [5]: (-1 * close_month).loc[month].nlargest(2) *-1

Out[5]: A    1
        F    2
        Name: 2018-02-01 00:00:00, dtype: int64
```

Now you've seen how to get the top and bottom performing prices in a single month. Let's see if you can apply this knowledge. ## Quiz Implement `date_top_industries` to find the top performing closing prices and return their sectors for a single date. The function should only return the set of sectors, there shouldn't be any duplicates returned.

- The number of top performing prices to look at is represented by the parameter `top_n`.
- The `date` parameter is the date to look for the top performing prices in the `prices` DataFrame.
- The sector information for each ticker is located in the `sector` parameter.

For example:

```
                Prices
              A         B         C         D         E
2013-07-08    2         2         7         2         6
```

```
2013-07-09     5         3         6         7         5
...            ...       ...       ...

           Sector
A       "Utilities"
B       "Health Care"
C       "Real Estate"
D       "Real Estate"
E       "Information Technology"

Date:  2013-07-09
Top N: 3
```

The set created from the function `date_top_industries` should be the following:

```
{"Utilities", "Real Estate"}
```

Note: Stock A and E have the same price for the date, but only A's sector got returned. We'll keep it simple and only take the first occurrences of ties.

```
In [6]: import project_tests
```

```
        def date_top_industries(prices, sector, date, top_n):
            """
            Get the set of the top industries for the date

            Parameters
            ----------
            prices : DataFrame
                Prices for each ticker and date
            sector : Series
                Sector name for each ticker
            date : Date
                Date to get the top performers
            top_n : int
                Number of top performers to get

            Returns
            -------
            top_industries : set
                Top industries for the date
            """
            # TODO: Implement Function

            return set(sector[prices.loc[date].nlargest(top_n).index])
```

```
project_tests.test_date_top_industries(date_top_industries)
```

```
Tests Passed
```

1.1 Quiz Solution

If you're having trouble, you can check out the quiz solution here.