Part 1: Data scraping and preparation

Task 1: Scrape your competitor's data (10 pts)

Scrape data for the top 50 solar flares shown in SpaceWeatherLive.com (Links to an external site.).

Steps to do this are (if you are using python):

1. *pip install* or *conda install* the following Python packages: *beautifulsoup4, requests, pandas, NumPy, matplotlib* (for visualization)

   using Anaconda (Jupiter notebook) it is preloaded with all the packages.

2. Use *requests* to get page content (as in, HTTP GET)

```
In [3]:
import requests
url = "https://www.spaceweatherlive.com/en/solar-activity/top-50-solar-flares.html"
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.266
page = requests.get(url, headers=headers)

page.status_code

Out[3]: 200
```

3. Extract the text from the page

```
In [2]: page.text

Out[2]: '<!doctype html>\n<html lang="en">\n    <head>\n        <title>Top 50 solar flares | Solar activity | SpaceWeatherLive.com</t
itle>\n        <meta charset="utf-8" />\n        <meta name="robots" content="index, follow, max-snippet:-1, max-image-previe
w:large, max-video-preview:-1">\n        <meta name="description" content="On this page you will find an overview of the stro
ngest solar flares since June 1996 together with links to more information in our archive and a v...">\n        <meta name="k
eywords" content="SpaceWeatherLive, Live, Aurora, Auroral activity, Aurora Australis, Aurora Borealis, northern lights, Solar
wind, Kp-index, Space Weather, Space Weather Updates, Aurora forecast, Space Weather Alerts, Solar activity, Solar flares, Su
nspots, Aurora alert, Auroral activity, The Sun, SDO, STEREO, EPAM, DSCOVR">\n        <!-- Facebook meta -->\n        <meta p
roperty="og:url" content="https://spaceweatherlive.com/en/solar-activity/top-50-solar-flares.html">\n        <meta property
="og:type" content="article">\n        <meta property="og:title" content="Top 50 solar flares | Solar activity" />\n        <
meta property="og:description" content="On this page you will find an overview of the strongest solar flares since June 1996
together with links to more information in our archive and a video (if available) of the event. This page is updated daily.">
\n        <meta property="og:image" content="https://www.spaceweatherlive.com/images/share_headers/default.jpg">\n        <me
ta property="og:site_name" content="SpaceWeatherLive.com">\n        <meta property="fb:app_id" content="344241918924880">\n
<meta property="fb:admins" content="615289796">\n        <meta property="fb:admins" content="100002307882367">\n        <!--
Twitter Card meta -->\n        <meta name="twitter:card" content="summary_large_image">\n        <meta name="twitter:site" co
ntent="@_SpaceWeather_">\n        <meta name="twitter:creator" content="@_SpaceWeather_">\n        <meta name="twitter:url" c
ontent="https://spaceweatherlive.com/en/solar-activity/top-50-solar-flares.html">\n        <meta name="twitter:title" content
="Top 50 solar flares | Solar activity">\n        <meta name="twitter:description" content="On this page you will find an ove
rview of the strongest solar flares since June 1996 together with links to more information in our archive and a video (if av
```

4. Use BeautifulSoup to read and parse the data, either as html or lxml

```
In [4]: from bs4 import BeautifulSoup
        soup = BeautifulSoup(page.content, 'html.parser')
```

5. Use prettify() to view the content and find the appropriate table

```
In [11]: print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>
   Top 50 solar flares | Solar activity | SpaceWeatherLive.com
  </title>
  <meta charset="utf-8"/>
  <meta content="index, follow, max-snippet:-1, max-image-preview:large, max-video-preview:-1" name="robots"/>
  <meta content="On this page you will find an overview of the strongest solar flares since June 1996 together w
more information in our archive and a v..." name="description"/>
  <meta content="SpaceWeatherLive, Live, Aurora, Auroral activity, Aurora Australis, Aurora Borealis, northern l
wind, Kp-index, Space Weather, Space Weather Updates, Aurora forecast, Space Weather Alerts, Solar activity, Sol
nspots, Aurora alert, Auroral activity, The Sun, SDO, STEREO, EPAM, DSCOVR" name="keywords"/>
  <!-- Facebook meta -->
  <meta content="https://spaceweatherlive.com/en/solar-activity/top-50-solar-flares.html" property="og:url"/>
  <meta content="article" property="og:type"/>
  <meta content="Top 50 solar flares | Solar activity" property="og:title">
   <meta content="On this page you will find an overview of the strongest solar flares since June 1996 together
more information in our archive and a video (if available) of the event. This page is updated daily." property="
```

6. Use find() to save the aforementioned table as a variable

```
In [12]: soup.find("table", attrs={"class": "table table-striped"})
```

```
Out[12]: <table class="table table-striped"><thead><tr><th></th><th></th><th></th><th>Region</th><th>Start</th><th>Maximum<
</th><th></th></tr></thead><tbody><tr><td>1</td><td><span class="flare_small Xclass">X28+</span></td><td>2003/11/0
a class="btn btn-SWL RegionBGD" href="/en/solar-activity/region/10486.html">0486</a></td><td>19:29</td><td>19:53</
6</td><td><div class="btn-group"><a class="btn btn-xs btn-outline-dark" data-fancybox="" href="/images/Archief/200
0311041953X28.0.mp4"><i aria-hidden="true" class="far fa-play-circle"></i> Movie</a><a class="btn btn-xs btn-outli
ef="/en/archive/2003/11/04/xray.html">View archive</a></div></td></tr><tr><td>2</td><td><span class="flare_small X
</span></td><td>2001/04/02</td><td><a class="btn btn-SWL RegionBGD" href="/en/solar-activity/region/9393.html">939
td>21:32</td><td>21:51</td><td>22:03</td><td><div class="btn-group"><a class="btn btn-xs btn-outline-dark" data-fa
ref="/images/Archief/2001/Flares/200104022151X20.0.mp4"><i aria-hidden="true" class="far fa-play-circle"></i> Movi
ss="btn btn-xs btn-outline-dark" href="/en/archive/2001/04/02/xray.html">View archive</a></div></td></tr><tr><td>3
pan class="flare_small Xclass">X17.2+</span></td><td>2003/10/28</td><td><a class="btn btn-SWL RegionBGD" href="/en
vity/region/10486.html">0486</a></td><td>09:51</td><td>11:10</td><td>11:24</td><td><div class="btn-group"><a class
s btn-outline-dark" data-fancybox="" href="/images/Archief/2003/Flares/200310281110X17.2.mp4"><i aria-hidden="true
r fa-play-circle"></i> Movie</a><a class="btn btn-xs btn-outline-dark" href="/en/archive/2003/10/28/xray.html">Vie
a></div></td></tr><tr><td>4</td><td><span class="flare_small Xclass">X17+</span></td><td>2005/09/07</td><td><a cla
-SWL Region" href="/en/solar-activity/region/10808.html">0808</a></td><td>17:17</td><td>17:40</td><td>18:03</td><t
s="btn-group"><a class="btn btn-xs btn-outline-dark" data-fancybox="" href="/images/Archief/2005/Flares/2005090717
4"><i aria-hidden="true" class="far fa-play-circle"></i> Movie</a><a class="btn btn-xs btn-outline-dark" href="/en
05/09/07/xray.html">View archive</a></div></td></tr><tr><td>5</td><td><span class="flare_small Xclass">X14.4</span
```

```
table = soup.find("table", attrs={"class": "table table-striped"})
table_body = table.find('tbody')
data=[]
rows = table_body.find_all('tr')
for row in rows:
    cols = row.find_all('td')
    cols = [ele.text.strip() for ele in cols]
    data.append([ele for ele in cols if ele])
print(data)
```

```
[['1', 'X28+', '2003/11/04', '0486', '19:29', '19:53', '20:06', 'MovieView archive'], ['2', 'X20+', '2001/04/02', '9393',
2', '21:51', '22:03', 'MovieView archive'], ['3', 'X17.2+', '2003/10/28', '0486', '09:51', '11:10', '11:24', 'MovieView a
e'], ['4', 'X17+', '2005/09/07', '0808', '17:17', '17:40', '18:03', 'MovieView archive'], ['5', 'X14.4', '2001/04/15', '9
'13:19', '13:50', '13:55', 'MovieView archive'], ['6', 'X10', '2003/10/29', '0486', '20:37', '20:49', '21:01', 'MovieView
ve'], ['7', 'X9.4', '1997/11/06', '8100', '11:49', '11:55', '12:01', 'MovieView archive'], ['8', 'X9.3', '2017/09/06', '2
'11:53', '12:02', '12:10', 'MovieView archive'], ['9', 'X9', '2006/12/05', '0930', '10:18', '10:35', '10:45', 'MovieView
e'], ['10', 'X8.3', '2003/11/02', '0486', '17:03', '17:25', '17:39', 'MovieView archive'], ['11', 'X8.2', '2017/09/10',
'15:35', '16:06', '16:31', 'MovieView archive'], ['12', 'X7.1', '2005/01/20', '0720', '06:36', '07:01', '07:26', 'MovieVi
hive'], ['13', 'X6.9', '2011/08/09', '1263', '07:48', '08:05', '08:08', 'MovieView archive'], ['14', 'X6.5', '2006/12/06'
0', '18:29', '18:47', '19:00', 'MovieView archive'], ['15', 'X6.2', '2005/09/09', '0808', '19:13', '20:04', '20:36', 'Mov
archive'], ['16', 'X6.2', '2001/12/13', '9733', '14:20', '14:30', '14:35', 'MovieView archive'], ['17', 'X5.7', '2000/07/
'9077', '10:03', '10:24', '10:43', 'MovieView archive'], ['18', 'X5.6', '2001/04/06', '9415', '19:10', '19:21', '19:31',
View archive'], ['19', 'X5.4', '2012/03/07', '1429', '00:02', '00:24', '00:40', 'MovieView archive'], ['20', 'X5.4', '200
8', '0808', '20:52', '21:06', '21:17', 'MovieView archive'], ['21', 'X5.4', '2003/10/23', '0486', '08:19', '08:35', '08:4
ovieView archive'], ['22', 'X5.3', '2001/08/25', '9591', '16:23', '16:45', '17:04', 'MovieView archive'], ['23', 'X4.9',
02/25', '1990', '00:39', '00:49', '01:03', 'MovieView archive'], ['24', 'X4.9', '1998/08/18', '8307', '22:10', '22:19',
8', 'View archive'], ['25', 'X4.8', '2002/07/23', '0039', '00:18', '00:35', '00:47', 'MovieView archive'], ['26', 'X4',
1/26', '9236', '16:34', '16:48', '16:56', 'MovieView archive'], ['27', 'X3.9', '2003/11/03', '0488', '09:43', '09:55',
'MovieView archive'], ['28', 'X3.9', '1998/08/19', '8307', '21:35', '21:45', '21:50', 'View archive'], ['29', 'X3.8', '20
17', '0720', '06:59', '09:52', '10:07', 'MovieView archive'], ['30', 'X3.7', '1998/11/22', '8384', '06:30', '06:42', '06:
'MovieView archive'], ['31', 'X3.6', '2005/09/09', '0808', '09:42', '09:59', '10:08', 'MovieView archive'], ['32', 'X3.6'
4/07/16', '0649', '13:49', '13:55', '14:01', 'MovieView archive'], ['33', 'X3.6', '2003/05/28', '0365', '00:17', '00:27',
9', 'MovieView archive'], ['34', 'X3.4', '2006/12/13', '0930', '02:14', '02:40', '02:57', 'MovieView archive'], ['35', 'X
'2001/12/28', '9767', '20:02', '20:45', '21:32', 'MovieView archive'], ['36', 'X3.3', '2013/11/05', '1890', '22:07', '22:
'22:15', 'MovieView archive'], ['37', 'X3.3', '2002/07/20', '0039', '21:04', '21:30', '21:54', 'MovieView archive'], ['38
3.3', '1998/11/28', '8395', '04:54', '05:52', '06:13', 'MovieView archive'], ['39', 'X3.2', '2013/05/14', '1748', '00:00'
```

7. Use pandas to read in the HTML file. HINT make-sure the above data is properly typecast.

In [40]:
```
import pandas as pd

table = soup.find("table", attrs={"class": "table table-striped"})
table_body = table.find('tbody')
res=[]
table_rows = table_body.find_all('tr')
for tr in table_rows:
    td= tr.find_all('td')
    row= [tr.text.strip() for tr in td if tr.text.strip()]
    if row:
        res.append(row)

df = pd.DataFrame(res, columns=[" rank ", "x_classification", " date ", " region ", " start_time ", " maximum_time ",
                                " end_time " , " movie "])
df.head(50)
```

8. Set reasonable names for the table columns, e.g., rank, x_classification, date, region, start_time, maximum_time, end_time, movie. *Pandas.columns* makes this very simple.

	rank	x_classification	date	region	start_time	maximum_time	end_time	movie
0	1	X28+	2003/11/04	0486	19:29	19:53	20:06	MovieView archive
1	2	X20+	2001/04/02	9393	21:32	21:51	22:03	MovieView archive
2	3	X17.2+	2003/10/28	0486	09:51	11:10	11:24	MovieView archive
3	4	X17+	2005/09/07	0808	17:17	17:40	18:03	MovieView archive
4	5	X14.4	2001/04/15	9415	13:19	13:50	13:55	MovieView archive
5	6	X10	2003/10/29	0486	20:37	20:49	21:01	MovieView archive
6	7	X9.4	1997/11/06	8100	11:49	11:55	12:01	MovieView archive
7	8	X9.3	2017/09/06	2673	11:53	12:02	12:10	MovieView archive
8	9	X9	2006/12/05	0930	10:18	10:35	10:45	MovieView archive
9	10	X8.3	2003/11/02	0486	17:03	17:25	17:39	MovieView archive
10	11	X8.2	2017/09/10	2673	15:35	16:06	16:31	MovieView archive
11	12	X7.1	2005/01/20	0720	06:36	07:01	07:26	MovieView archive
12	13	X6.9	2011/08/09	1263	07:48	08:05	08:08	MovieView archive
13	14	X6.5	2006/12/06	0930	18:29	18:47	19:00	MovieView archive
14	15	X6.2	2005/09/09	0808	19:13	20:04	20:36	MovieView archive
15	16	X6.2	2001/12/13	9733	14:20	14:30	14:35	MovieView archive
16	17	X5.7	2000/07/14	9077	10:03	10:24	10:43	MovieView archive
17	18	X5.6	2001/04/06	9415	19:10	19:21	19:31	MovieView archive
18	19	X5.4	2012/03/07	1429	00:02	00:24	00:40	MovieView archive

19	20	X5.4	2005/09/08	0808	20:52	21:06	21:17	MovieView archive
20	21	X5.4	2003/10/23	0486	08:19	08:35	08:49	MovieView archive
21	22	X5.3	2001/08/25	9591	16:23	16:45	17:04	MovieView archive
22	23	X4.9	2014/02/25	1990	00:39	00:49	01:03	MovieView archive
23	24	X4.9	1998/08/18	8307	22:10	22:19	22:28	View archive
24	25	X4.8	2002/07/23	0039	00:18	00:35	00:47	MovieView archive
25	26	X4	2000/11/26	9236	16:34	16:48	16:56	MovieView archive
26	27	X3.9	2003/11/03	0488	09:43	09:55	10:19	MovieView archive
27	28	X3.9	1998/08/19	8307	21:35	21:45	21:50	View archive
28	29	X3.8	2005/01/17	0720	06:59	09:52	10:07	MovieView archive
29	30	X3.7	1998/11/22	8384	06:30	06:42	06:49	MovieView archive
30	31	X3.6	2005/09/09	0808	09:42	09:59	10:08	MovieView archive
31	32	X3.6	2004/07/16	0649	13:49	13:55	14:01	MovieView archive
32	33	X3.6	2003/05/28	0365	00:17	00:27	00:39	MovieView archive
33	34	X3.4	2006/12/13	0930	02:14	02:40	02:57	MovieView archive
34	35	X3.4	2001/12/28	9767	20:02	20:45	21:32	MovieView archive
35	36	X3.3	2013/11/05	1890	22:07	22:12	22:15	MovieView archive
36	37	X3.3	2002/07/20	0039	21:04	21:30	21:54	MovieView archive
37	38	X3.3	1998/11/28	8395	04:54	05:52	06:13	MovieView archive
38	39	X3.2	2013/05/14	1748	00:00	01:11	01:20	MovieView archive
39	40	X3.1	2014/10/24	2192	21:07	21:41	22:13	MovieView archive
40	41	X3.1	2002/08/24	0069	00:49	01:12	01:31	MovieView archive
41	42	X3	2002/07/15	0030	19:59	20:08	20:14	MovieView archive
42	43	X2.8	2013/05/13	1748	15:48	16:05	16:16	MovieView archive
43	44	X2.8	2001/12/11	9733	07:58	08:08	08:14	MovieView archive
44	45	X2.8	1998/08/18	8307	08:14	08:24	08:32	View archive
45	46	X2.7	2015/05/05	2339	22:05	22:11	22:15	MovieView archive
46	47	X2.7	2003/11/03	0488	01:09	01:30	01:45	MovieView archive
47	48	X2.7	1998/05/06	8210	07:58	08:09	08:20	MovieView archive
48	49	X2.6	2005/01/15	0720	22:25	23:02	23:31	MovieView archive
49	50	X2.6	2001/09/24	9632	09:32	10:38	11:09	MovieView archive

Task 2: Tidy the top 50 solar flare data (10 pts)

Make sure this table is usable using pandas:

1. Drop the last column of the table, since we are not going to use it moving forward.
2. Use datetime import to combine the date and each of the three time columns into three datetime columns. You will see why this is useful later on. iterrows() should prove useful here.
3. Update the values in the dataframe as you do this. Set_value should prove useful.
4. Set regions coded as - as missing (NaN). You can use dataframe.replace() here.

```
import pandas as pd
import numpy as numpy

table = soup.find("table", attrs={"class": "table table-striped"})
table_body = table.find('tbody')
res=[]
table_rows = table_body.find_all('tr')
for tr in table_rows:
    td= tr.find_all('td')
    row= [tr.text.strip() for tr in td if tr.text.strip()]
    if row:
        res.append(row)

df = pd.DataFrame(res, columns=["rank", "x_classification", "date", "region", "start_time", "maximum_time",
                                "end_time" , "movie"])

df.dtypes

# Combine the date column with the time columns to convert to datetime format
df_st = pd.to_datetime(df['date'] + ' ' + df['start_time'])
df_mt = pd.to_datetime(df['date'] + ' ' + df['maximum_time'])
df_et = pd.to_datetime(df['date'] + ' ' + df['end_time'])

# Add the new columns of the datetimes
df['start_datetime'] = df_st
df['max_datetime'] = df_mt
df['end_datetime'] = df_et

# Drop the unneeded columns
df = df.drop('date', 1)
df = df.drop('start_time', 1)
df = df.drop('maximum_time', 1)
df = df.drop('end_time', 1)
```

```
# Change the order of the columns
df = df[['rank', 'x_classification', 'start_datetime', 'max_datetime', 'end_datetime', 'region']]

# Replace all instances of '-' with '<NA>' as stated by the prompt
df = df.replace('-', 'NAN')

df
```

	rank	x_classification	start_datetime	max_datetime	end_datetime	region
0	1	X28+	2003-11-04 19:29:00	2003-11-04 19:53:00	2003-11-04 20:06:00	0486
1	2	X20+	2001-04-02 21:32:00	2001-04-02 21:51:00	2001-04-02 22:03:00	9393
2	3	X17.2+	2003-10-28 09:51:00	2003-10-28 11:10:00	2003-10-28 11:24:00	0486
3	4	X17+	2005-09-07 17:17:00	2005-09-07 17:40:00	2005-09-07 18:03:00	0808
4	5	X14.4	2001-04-15 13:19:00	2001-04-15 13:50:00	2001-04-15 13:55:00	9415
5	6	X10	2003-10-29 20:37:00	2003-10-29 20:49:00	2003-10-29 21:01:00	0486
6	7	X9.4	1997-11-06 11:49:00	1997-11-06 11:55:00	1997-11-06 12:01:00	8100
7	8	X9.3	2017-09-06 11:53:00	2017-09-06 12:02:00	2017-09-06 12:10:00	2673
8	9	X9	2006-12-05 10:18:00	2006-12-05 10:35:00	2006-12-05 10:45:00	0930
9	10	X8.3	2003-11-02 17:03:00	2003-11-02 17:25:00	2003-11-02 17:39:00	0486
10	11	X8.2	2017-09-10 15:35:00	2017-09-10 16:06:00	2017-09-10 16:31:00	2673
11	12	X7.1	2005-01-20 06:36:00	2005-01-20 07:01:00	2005-01-20 07:26:00	0720

Task 3: Scrape the NASA data (15 pts)

Next, you need to scrape NASA data (Links to an external site.) to get additional features about these solar flares. This table format is described here. (Links to an external site.)

Once scraped, do the next steps:

1. Use BeautifulSoup functions (e.g., find, findAll) and string functions (e.g., split and built-in slicing capabilities) to obtain each row of data as a long string.
2. Use the split function to separate each line of text into a data row.
3. Create a DataFrame with the data from the table.
4. Choose appropriate names for columns.

The result of this step should be similar to:

Dimension: 482 × 14

In [7]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup

r = requests.get('https://cdaw.gsfc.nasa.gov/CME_list/radio/waves_type2.html')
rt = BeautifulSoup(r.content,'html5lib')
table = rt.find('pre')
row_marker = 0
content = table.get_text()
lines = content.split('\n')

for i in range(0,11):
    lines.pop(0);
lines.pop(len(lines)-1)
lines.pop(len(lines)-1)
lines.pop(len(lines)-1)
dfNASA = pd.DataFrame(columns=['start_date', 'start_time', 'end_date', 'end_time', 'start_frequency',
                               'end_frequency', 'flare_location', 'flare_region',
                               'flare_classification', 'cme_date', 'cme_time', 'cme_angle', 'cme_width', 'cme_speed'],
                      index = range(0,len(lines)))
for line in lines:
    cols = line.split(' ')
    while '' in cols:
        cols.remove('')
    col_marker = 0
    while col_marker < 14:
        dfNASA.iat[row_marker, col_marker] = cols[col_marker]
        col_marker += 1
    row_marker += 1
```

Out[20]:

	start_date	start_time	end_date	end_time	start_frequency	end_frequency	flare_location	flare_region	flare_classification	cme_date	cme_time	cme_angl
0	1997/04/01	14:00	04/01	14:15	8000	4000	S25E16	8026	M1.3	04/01	15:18	7
1	1997/04/07	14:30	04/07	17:30	11000	1000	S28E19	8027	C6.8	04/07	14:27	Hal
2	1997/05/12	05:15	05/14	16:00	12000	80	N21W08	8038	C1.3	05/12	05:30	Hal
3	1997/05/21	20:20	05/21	22:00	5000	500	N05W12	8040	M1.3	05/21	21:00	26
4	1997/09/23	21:53	09/23	22:16	6000	2000	S29E25	8088	C1.4	09/23	22:02	13
...	...	...	...	...	...	...	...	...	...	...	...	...
516	2017/09/12	07:38	09/12	07:43	16000	13000	N08E48	12680	C3.0	09/12	08:03	12
517	2017/09/17	11:45	09/17	12:35	16000	900	S08E170	-----	----	09/17	12:00	Hal
518	2017/10/18	05:48	10/18	12:40	16000	400	S06E123	-----	----	10/18	08:00	8
519	2019/05/03	23:52	05/04	00:16	13000	2300	N12E82	12740	C1.0	05/03	23:24	9
520	2020/11/29	13:07	11/29	15:23	14000	850	S23E89	-----	M4.4	11/29	13:25	Hal

521 rows × 14 columns

Dimension: 521 × 14

Task 4: Tidy the NASA table (15 pts)

Here we will code missing observations properly, recode columns that correspond to more than one piece of information, and treat dates and times appropriately.

1. Recode any missing entries as NaN. Refer to the data description (Links to an external site.) to see how missing entries are encoded in each column. Be sure to look carefully at the actual data, as the nasa descriptions might not be completely accurate.
2. The CPA column (cme_angle) contains angles in degrees for most rows, except for halo flares, which are coded as Halo. Create a new column that indicates if a row corresponds to a halo flare or not, and then replace Halo entries in the cme_angle column with NaN.

3. The width column indicates if the given value is a lower bound. Create a new column that indicates if width is given as a lower bound, and remove any non-numeric part of the width column.
4. Combine date and time columns for start, end and cme so they can be encoded as datetime objects.

```
dfNASA_st = pd.to_datetime(dfNASA['start_date'] + ' ' + dfNASA['start_time'],errors='coerce')
dfNASA_et = pd.to_datetime(dfNASA['start_date'] + ' ' + dfNASA['end_time'],errors='coerce')
dfNASA_cme = pd.to_datetime(dfNASA['start_date'] + ' ' + dfNASA['cme_time'],errors='coerce')
# Add the new columns of the datetimes
dfNASA['start_datetime'] = dfNASA_st
dfNASA['cme_datetime'] = dfNASA_cme
dfNASA['end_datetime'] = dfNASA_et

# Drop the unneeded columns
dfNASA = dfNASA.drop('start_date', 1)
dfNASA = dfNASA.drop('start_time', 1)
dfNASA = dfNASA.drop('end_date', 1)
dfNASA = dfNASA.drop('end_time', 1)
dfNASA = dfNASA.drop('cme_date', 1)
dfNASA = dfNASA.drop('cme_time', 1)
# Replace all instances of '-' with '<NA>' as stated by the prompt
dfNASA = dfNASA.replace('----', 'NAN')
dfNASA = dfNASA.replace('-----', 'NAN')
dfNASA = dfNASA.replace('NaT', 'NAN')
#The CPA column (cme_angle) contains angles in degrees for most rows, except for halo flares, which are coded as Halo.
#Create a new column that indicates if a row corresponds to a halo
#flare or not, and then replace Halo entries in the cme_angle column with NaN
dfNASA.loc[dfNASA['cme_angle'] == 'Halo', 'is_halo'] = 'True'
dfNASA.loc[dfNASA['cme_angle'] != 'Halo', 'is_halo'] = 'False'

#The width column indicates if the given value is a lower bound.
#Create a new column that indicates if width is given as a lower bound, and remove any non-numeric part of the width column.

dfNASA.loc[dfNASA['cme_width'].str.contains('>'),'width_lower_bound']='True'
dfNASA.loc[~dfNASA["cme_width"].str.contains('>'),'width_lower_bound']='False'
dfNASA['cme_width'] = dfNASA['cme_width'].str.extract('(\d+)', expand=False)

# Change the order of the columns
dfNASA = dfNASA[['start_datetime', 'end_datetime', 'start_frequency', 'end_frequency', 'flare_location', 'flare_region',
        'flare_classification'
       ,'cme_datetime','cme_width','is_halo','width_lower_bound']]

# Sort rows by flare class
dfNASA = dfNASA.sort_values('flare_classification', ascending = False)
```

	start_datetime	end_datetime	start_frequency	end_frequency	flare_location	flare_region	flare_classification	cme_datetime	cme_width	is_halo	width_low
8	1997-11-06 12:20:00	1997-11-06 08:30:00	14000	100	S18W63	8100	X9.4	1997-11-06 12:10:00	360	True	
514	2017-09-06 12:05:00	2017-09-06 08:00:00	16000	70	S08W33	12673	X9.3	2017-09-06 12:24:00	360	True	
328	2006-12-05 10:50:00	2006-12-05 20:00:00	14000	250	S07E68	10930	X9.0	NaT	NaN	False	
515	2017-09-10 16:02:00	2017-09-10 06:50:00	16000	150	S09W92	NAN	X8.3	2017-09-10 16:00:00	360	True	
237	2003-11-02 17:30:00	2003-11-02 01:00:00	12000	250	S14W56	10486	X8.3	2003-11-02 17:30:00	360	True	
...	...	...	...	...	...	...	...	...	...	...	
10	1997-12-12 22:45:00	1997-12-12 23:20:00	14000	8000	N25W52	8116	B9.4	1997-12-12 00:26:00	73	False	
21	1998-05-11 21:40:00	1998-05-11 22:00:00	10000	1000	N32W90	8214	B6.6	1998-05-11 21:55:00	301	False	
22	1998-05-19 10:00:00	1998-05-19 11:30:00	14000	3000	N23W43	8222	B5.7	1998-05-19 10:27:00	139	False	
410	2013-08-06 02:01:00	2013-08-06 02:11:00	14000	11000	N27E25	EP	B4.5	2013-08-06 02:12:00	207	False	
338	2008-04-26 14:23:00	2008-04-26 14:39:00	7600	4900	N08E09	NAN	B3.8	2008-04-26 14:30:00	281	False	

521 rows × 11 columns

Dimension: 521 × 11

Part 2: Analysis

Now that you have data from both sites, let's start some analysis.

Task 5: Replication (10 pts)

Can you replicate the top 50 solar flare table in SpaceWeatherLive.com exactly using the data obtained from NASA? That is if you get the top 50 solar flares from the NASA table based on their classification (e.g., X28 is the highest), do you get data for the same solar flare events? Include code used to get the top 50 solar flares from the NASA table (be careful when ordering by classification). Write a sentence or two discussing how well you can replicate the SpaceWeatherLive data from the NASA data.

Solution:

Yes we can do that and here is my strategy:

1. Get the rows that have X class flare.
2. Get rid of X from the above dataframe for sorting purposes
3. Change type to float
4. Sort rows by flare class[top 50 solar flares from the NASA table based on their classification (e.g., X28 is the highest)

5. Extract the top 50 as 50Flare has 50 records
6. Put back the X in the flare_classification column values(concate)

Below is the Code snipped :

```
#Step 1: Get the rows that have X class flare
DfBF = dfNASA.loc[dfNASA['flare_classification'].str.contains('X')]
# Get rid of X for sorting purposes
DfBF['flare_classification'] = DfBF['flare_classification'].str.lstrip('X')

#Step 2: Change type to float
DfBF['flare_classification'] = DfBF.flare_classification.astype(float)

#Step3 : Sort rows by flare class[top 50 solar flares from the NASA table based on their classification (e.g., X28 is the highest
DfBF = DfBF.sort_values('flare_classification', ascending = False)

#Step 4: Extract the top 50 as 50Flare has 50 records
DfBF = DfBF.head(50)

#Step 5: Put back the X in the flare_classification column values(concate)
DfBF['flare_classification'] = DfBF.flare_classification.astype(str)
DfBF['flare_classification'] = "X" + DfBF['flare_classification']
DfBF
```

	start_datetime	end_datetime	start_frequency	end_frequency	flare_location	flare_region	flare_classification	cme_datetime	cme_width	is_halo	width_lower_bound
240	2003-11-04 20:00:00	NaT	10000	200	S19W83	10486	X28.0	2003-11-04 19:54:00	360	True	False
117	2001-04-02 22:05:00	2001-04-02 02:30:00	14000	250	N19W72	9393	X20.0	2001-04-02 22:06:00	244	False	False
233	2003-10-28 11:10:00	NaT	14000	40	S16E08	10486	X17.0	2003-10-28 11:30:00	360	True	False
126	2001-04-15 14:05:00	2001-04-15 13:00:00	14000	40	S20W85	9415	X14.0	2001-04-15 14:06:00	167	False	False
234	2003-10-29 20:55:00	NaT	11000	500	S15W02	10486	X10.0	2003-10-29 20:54:00	360	True	False
8	1997-11-06 12:20:00	1997-11-06 08:30:00	14000	100	S18W63	8100	X9.4	1997-11-06 12:10:00	360	True	False
514	2017-09-06 12:05:00	2017-09-06 08:00:00	16000	70	S08W33	12673	X9.3	2017-09-06 12:24:00	360	True	False
	2006-12-05	2006-12-05									

Task 6: Integration (15 pts)

Write a function that finds the best matching row in the NASA data for each of the top 50 solar flares in the SpaceWeatherLive data. Here, you have to decide for yourself how you determine what is the best matching entry in the NASA data for each of the top 50 solar flares. In your submission, include an explanation of how you are defining best matching rows across the two datasets in addition to the code used to find the best

matches. Finally, use your function to add a new column to the NASA dataset indicating its rank according to SpaceWeatherLive if it appears in that dataset.

Solution:

I have joined the table of the top 50 with large NASA solar flare table based off of which entities shared the same main classification.

I tried to join X_classification with Flare_Classification but it cannot be done because there are discrepancies between how the two data sources classified. I split this column into two and made main_classification and sub_Classification. for example: if x_classification has value as x2.6 the main_classification will be x2 and subclassification will be 6.if a column does not have sub_Classification it will be 0 value.

but this join condition was not enough.as I was getting two different ranking for same classification.

```
In [9]: df_merge = pd.merge( DfBF,df , how='inner', left_on = ['flare_main_classification'],
                             right_on = ['x_main_classification']
                             )

        #df_merge = df_merge.sort_values('rank', ascending = False)

        df_merge
```

Out[9]:

	flare_classification	start_datetime_x	end_datetime_x	start_frequency	end_frequency	flare_location	flare_region	cme_datetime	cme_width	is_halo	...	ra
0	X28.0	2003-11-04 20:00:00	NaT	10000	200	S19W83	1048	2003-11-04 19:54:00	360	True	...	
1	X20.0	2001-04-02 22:05:00	2001-04-02 02:30:00	14000	250	N19W72	9393	2001-04-02 22:06:00	244	False	...	
2	X17.0	2003-10-28 11:10:00	NaT	14000	40	S16E08	1048	2003-10-28 11:30:00	360	True	...	
3	X17.0	2003-10-28 11:10:00	NaT	14000	40	S16E08	1048	2003-10-28 11:30:00	360	True	...	

After analyzing a bit more I found apart from this classification field we can use start_datetime field for our join condition. But here if we join two columns it will not match as although the date is same, **time** data is different so I split again the date and time into two different fields and joined the dataframes with start_date field. after this join we got 35 rows of matching record.

	flare_classification	start_datetime_x	rank
0	X28.0	2003-11-04 20:00:00	1
1	X20.0	2001-04-02 22:05:00	2
2	X17.0	2003-10-28 11:10:00	3
3	X14.0	2001-04-15 14:05:00	5
4	X10.0	2003-10-29 20:55:00	6
5	X9.4	1997-11-06 12:20:00	7
6	X9.3	2017-09-06 12:05:00	8
7	X9.0	2006-12-05 10:50:00	9
8	X8.3	2003-11-02 17:30:00	10
9	X8.3	2017-09-10 16:02:00	11
10	X7.1	2005-01-20 07:15:00	12
11	X6.9	2011-08-09 08:20:00	13
12	X6.5	2006-12-06 18:00:00	14
13	X6.2	2005-09-09 19:45:00	15
14	X5.7	2000-07-14 10:30:00	17
15	X5.6	2001-04-06 19:35:00	18
16	X5.4	2012-03-07 01:00:00	19
17	X5.3	2001-08-25 16:50:00	22
18	X4.9	2014-02-25 00:56:00	23
19	X4.8	2002-07-23 00:50:00	25
20	X4.0	2000-11-26 17:00:00	26
21	X3.9	2003-11-03 10:00:00	27
22	X3.8	2005-01-17 10:00:00	29
23	X3.6	2003-05-28 01:00:00	33
24	X3.4	2006-12-13 02:45:00	34
25	X3.4	2001-12-28 20:35:00	35
26	X3.3	2002-07-20 21:30:00	37
27	X3.2	2013-05-14 01:16:00	39
28	X3.1	2002-08-24 01:45:00	41
29	X2.8	2013-05-13 16:15:00	43
30	X2.7	2015-05-05 22:24:00	46
31	X2.7	1998-05-06 08:25:00	48
32	X2.7	2003-11-03 01:15:00	47
33	X2.6	2005-01-15 23:00:00	49
34	X2.6	2001-09-24 10:45:00	50

Here is the Function:

```
def IntegrateTwoDF(df,DfBF,DfBFfull):
    DfBF[['flare_main_classification','flare_sub_classification']] = DfBF['flare_classification'].str.split('.',expand=True)
    DfBF['flare_Start_date'] = [d.date() for d in DfBF['start_datetime']]
    DfBF['flare_Start_time'] = [d.time() for d in DfBF['start_datetime']]
    DfBF.flare_Start_date = pd.to_datetime(DfBF.flare_Start_date)


    #remove unwanter chacter from x_Classification
    df['x_classification'] = df['x_classification'].str.rstrip('+')
    #split x classification into two parts for data integation
    df[['x_main_classification','x_sub_classification']] = df['x_classification'].str.split('.',expand=True)
    df['x_sub_classification'] = df['x_sub_classification'].fillna(0)
    df['x_Start_date'] = [d.date() for d in df['start_datetime']]
    df['x_Start_time'] = [d.time() for d in df['start_datetime']]

    df.x_Start_date = pd.to_datetime(df.x_Start_date)

    df_merge = pd.merge( DfBF,df , how='inner', left_on = ['flare_main_classification','flare_Start_date'],
                    right_on = ['x_main_classification','x_Start_date']
                                    )
    DFRes = df_merge[['flare_classification','start_datetime_x','rank']]

    dftoshow=pd.merge( dfNASA,DFRes, how='left', left_on = ['flare_classification','start_datetime'],
                    right_on = ['flare_classification','start_datetime_x'])

    return dftoshow

IntegrateTwoDF(df,DfBF,dfNASA)
```

atetime	end_datetime	start_frequency	end_frequency	flare_location	flare_region	cme_datetime	cme_width	is_halo	width_lower_bound	start_datetime_x	rank
7-04-01 4:00:00	1997-04-01 14:15:00	8000	4000	S25E16	8026	1997-04-01 15:18:00	79	False	False	NaT	NaN
7-04-07 4:30:00	1997-04-07 17:30:00	11000	1000	S28E19	8027	1997-04-07 14:27:00	360	True	False	NaT	NaN
7-05-12 5:15:00	1997-05-12 16:00:00	12000	80	N21W08	8038	1997-05-12 05:30:00	360	True	False	NaT	NaN
7-05-21 0:20:00	1997-05-21 22:00:00	5000	500	N05W12	8040	1997-05-21 21:00:00	165	False	False	NaT	NaN
7-09-23 1:53:00	1997-09-23 22:16:00	6000	2000	S29E25	8088	1997-09-23 22:02:00	155	False	False	NaT	NaN
...	...	...	...	...	...	...	...	...	...	...	...
7-09-12 7:38:00	2017-09-12 07:43:00	16000	13000	N08E48	12680	2017-09-12 08:03:00	96	False	False	NaT	NaN
7-09-17 1:45:00	2017-09-17 12:35:00	16000	900	S08E170	NAN	2017-09-17 12:00:00	360	True	False	NaT	NaN
7-10-18 5:48:00	2017-10-18 12:40:00	16000	400	S06E123	NAN	2017-10-18 08:00:00	146	False	False	NaT	NaN
9-05-03 3:52:00	2019-05-03 00:16:00	13000	2300	N12E82	12740	2019-05-03 23:24:00	113	False	False	NaT	NaN
0-11-29 3:07:00	2020-11-29 15:23:00	14000	850	S23E89	NAN	2020-11-29 13:25:00	360	True	False	NaT	NaN

Task 7: Attributes visualization (7 pts)

Plot attributes in the NASA dataset (e.g., starting or ending frequencies, flare height or width) over time. Use graphical elements (e.g., text or points) to indicate flares in the top 50 flares.

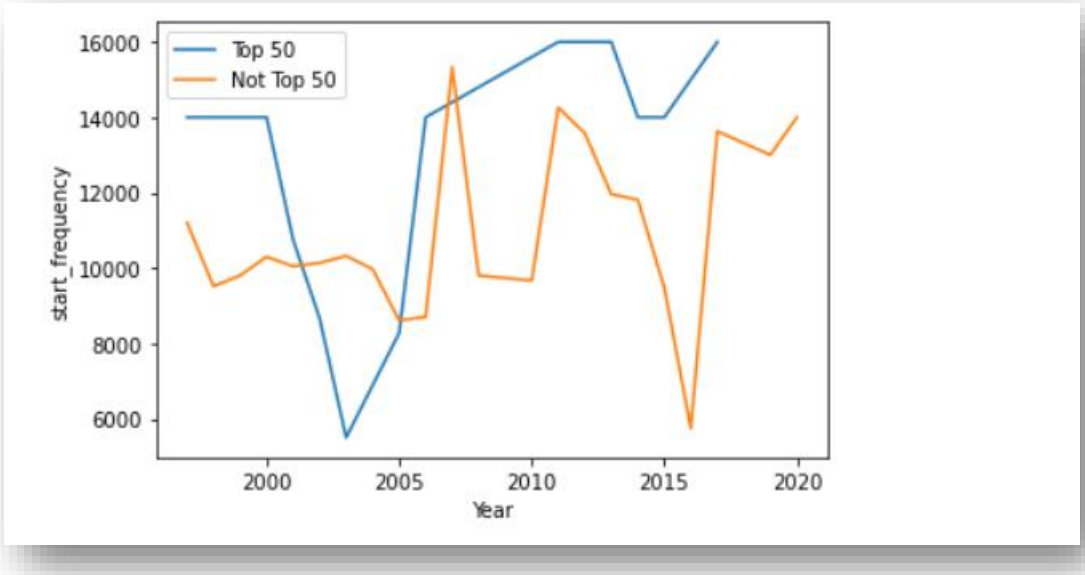
Solution:

- Plot attributes in the NASA dataset (Start_frequency) over time

```
NasaDF['Top_50'] = pd.notnull(NasaDF['rank'])
NasaDF.loc[NasaDF['Top_50'] == True, 'Top_50'] = 'Top_50'
NasaDF.loc[NasaDF['Top_50'] == False, 'Top_50'] = 'Not_Top_50'
NasaDF['year'] = NasaDF['start_datetime'].dt.year
NasaDF.year=pd.to_numeric(NasaDF.year)
NasaDF.start_frequency=pd.to_numeric(NasaDF.start_frequency,errors='coerce')
top50=NasaDF[NasaDF.Top_50=='Top_50']
Nottop50=NasaDF[NasaDF.Top_50=='Not_Top_50']

Newtop50 =  top50.groupby(['year'],as_index=False).start_frequency.mean()
Newnottop50 =  Nottop50.groupby(['year'],as_index=False).start_frequency.mean()

import matplotlib.pyplot as plt
plt.plot(Newtop50.year,Newtop50.start_frequency)
plt.plot(Newnottop50.year,Newnottop50.start_frequency)
plt.legend(['Top 50','Not Top 50'])
plt.xlabel('Year')
plt.ylabel('start_frequency')
plt.show()
```

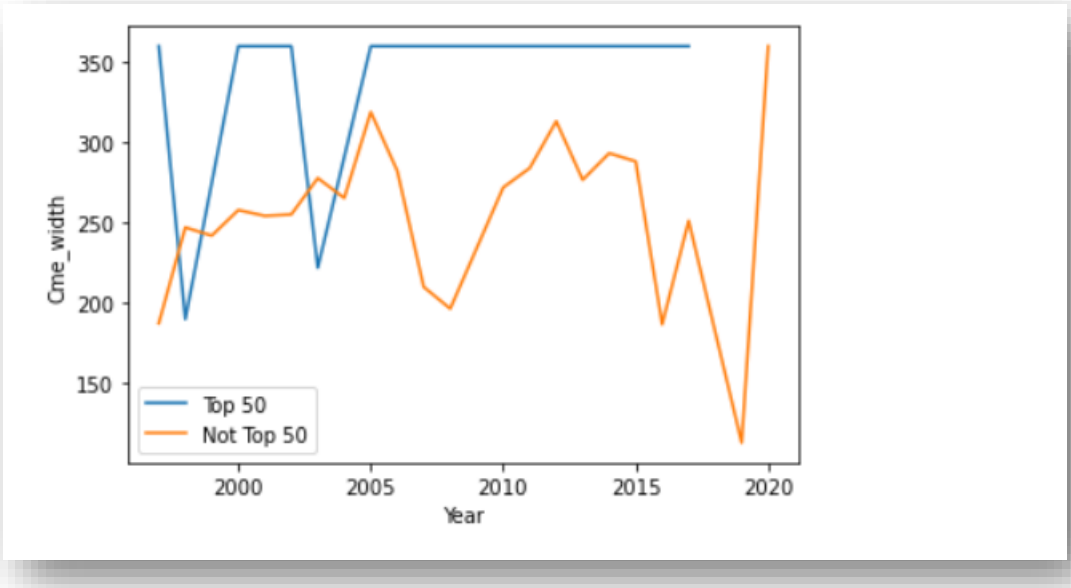


- Plot attributes in the NASA dataset (cme_width) over time

```
NasaDF['Top_50'] = pd.notnull(NasaDF['rank'])
NasaDF.loc[NasaDF['Top_50'] == True, 'Top_50'] = 'Top_50'
NasaDF.loc[NasaDF['Top_50'] == False, 'Top_50'] = 'Not_Top_50'
NasaDF['year'] = NasaDF['start_datetime'].dt.year
NasaDF.year=pd.to_numeric(NasaDF.year)
NasaDF.cme_width=pd.to_numeric(NasaDF.cme_width)
top50=NasaDF[NasaDF.Top_50=='Top_50']
Nottop50=NasaDF[NasaDF.Top_50=='Not_Top_50']

Newtop50 =  top50.groupby(['year'],as_index=False).cme_width.mean()
Newnottop50 =  Nottop50.groupby(['year'],as_index=False).cme_width.mean()

import matplotlib.pyplot as plt
plt.plot(Newtop50.year,Newtop50.cme_width)
plt.plot(Newnottop50.year,Newnottop50.cme_width)
plt.legend(['Top 50','Not Top 50'])
plt.xlabel('Year')
plt.ylabel('Cme_width')
plt.show()
```

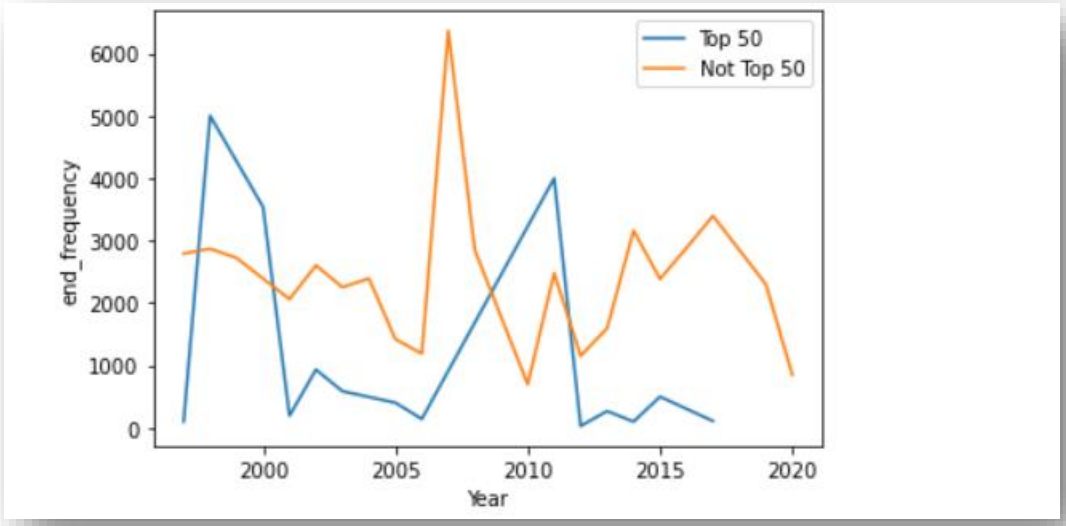


- Plot attributes in the NASA dataset (end_frequency) over time

```
NasaDF['Top_50'] = pd.notnull(NasaDF['rank'])
NasaDF.loc[NasaDF['Top_50'] == True, 'Top_50'] = 'Top_50'
NasaDF.loc[NasaDF['Top_50'] == False, 'Top_50'] = 'Not_Top_50'
NasaDF['year'] = NasaDF['start_datetime'].dt.year
NasaDF.year=pd.to_numeric(NasaDF.year)
NasaDF.end_frequency=pd.to_numeric(NasaDF.end_frequency,errors='coerce')
top50=NasaDF[NasaDF.Top_50=='Top_50']
Nottop50=NasaDF[NasaDF.Top_50=='Not_Top_50']

Newtop50 =  top50.groupby(['year'],as_index=False).end_frequency.mean()
Newnottop50 =  Nottop50.groupby(['year'],as_index=False).end_frequency.mean()

import matplotlib.pyplot as plt
plt.plot(Newtop50.year,Newtop50.end_frequency)
plt.plot(Newnottop50.year,Newnottop50.end_frequency)
plt.legend(['Top 50','Not Top 50'])
plt.xlabel('Year')
plt.ylabel('end_frequency')
plt.show()
```



Task 8: Attributes comparison (8 pts)

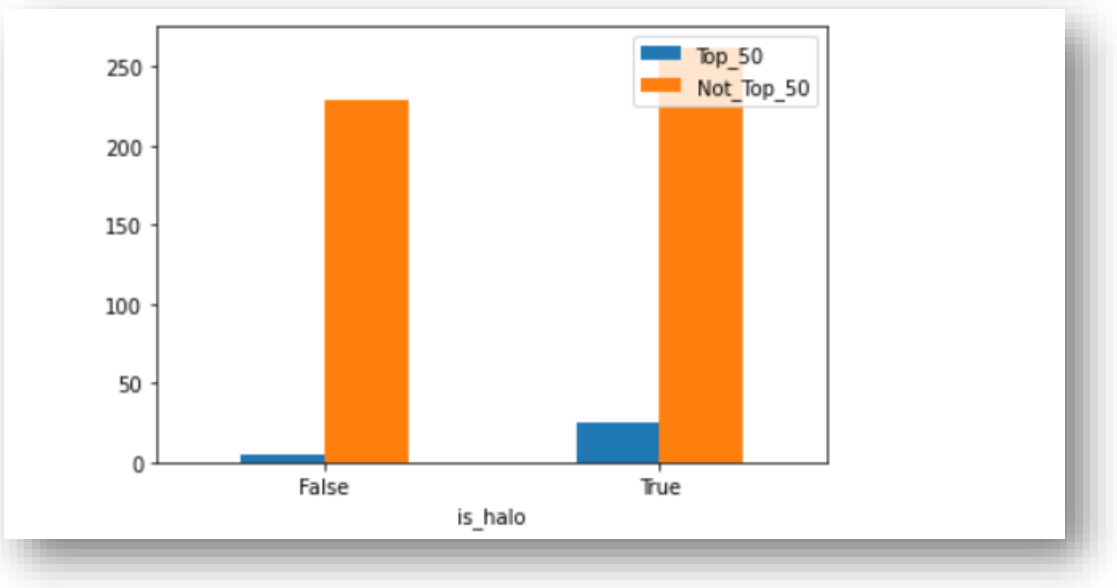
Do flares in the top 50 tend to have Halo CMEs? You can make a bar plot that compares the number (or proportion) of Halo CMEs in the top 50 flares vs. the dataset as a whole.

Solution:

```
grouped_df = NasaDF.groupby(['is_halo','Top_50']
                            ).size().reset_index(name="Count")
top50=grouped_df[grouped_df.Top_50=='Top_50']
Nottop50=grouped_df[grouped_df.Top_50=='Not_Top_50']

dftoshow=pd.merge( top50,Nottop50, how='inner', left_on = ['is_halo'],
                  right_on = ['is_halo'])
dftoshow['Top_50']=dftoshow.Count_x
dftoshow['Not_Top_50']=dftoshow.Count_y
dftoshow = dftoshow.drop('Count_x', 1)
dftoshow = dftoshow.drop('Count_y', 1)
dftoshow = dftoshow.drop('Top_50_x', 1)
dftoshow = dftoshow.drop('Top_50_y', 1)

dftoshow.plot.bar(x='is_halo',  rot=0)
```



Task 9: Events distribution (10 pts)

Do strong flares cluster in time? Plot the number of flares per month over time, add a graphical element to indicate (e.g., text or points) to indicate the number of strong flares (in the top 50) to see if they cluster.

Solution:

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
#Data Preparation for the plot
NasaDF['month_year'] = NasaDF['start_datetime'].dt.to_period('M')
Grouped_DF=NasaDF.groupby(['month_year','Top_50']).size().reset_index(name="Count")
top50=Grouped_DF[Grouped_DF.Top_50=='Top_50']
Nottop50=Grouped_DF[Grouped_DF.Top_50=='Not_Top_50']
top50 = top50.drop('Top_50', 1)
Nottop50 = Nottop50.drop('Top_50', 1)


fig = go.Figure()
fig.add_trace(
    go.Scatter(x=top50['month_year'].astype(dtype=str),
                        y=top50['Count'], name="Top 50"))
fig.add_trace(
    go.Scatter(x=Nottop50['month_year'].astype(dtype=str),
                        y=Nottop50['Count'], name="All Flares"))
fig.update_layout({"title": ' Number of flares per month over time',
                   "xaxis": {"title":"Year-Month"},
                   "yaxis": {"title":"number of flares"},
                   "showlegend": True})
fig.write_image("by-month.png",format="png", width=1000, height=600, scale=3)

fig.show()
```