Solar Flares

March 16, 2021

[1]:

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np
import datetime
import re
import functools
```

[ ]:

```

```

[2]:

```
# This block is for part 1 step 1
r = requests.get('https://cmsc320.github.io/files/top-50-solar-flares.html')

html_text=r.content

soup = BeautifulSoup(html_text, 'html.parser')
#soup.prettify()

# Just comment the above line since the output is long and messy

all_frames = pd.read_html(r.text,flavor='bs4')
using_table=all_frames[0]

using_table=using_table.rename(columns={"Unnamed: 0":"rank","Unnamed: 1":␣
 ↪"x_class","Unnamed: 2":"date"})
using_table.index=range(1,51)
using_table

using_table

# Here is the answer for Part 1 Step 1

# The code above is not very complicated.
# First, I use requests.get() to get the data from the website.

# Then, I find that pd.read_html can read the website itself, and the table we␣
 ↪need is the first
```

```
# one in the output.

# At last, I rename the column names.
```

```
[2]:     rank x_class        date  Region  Start Maximum    End         Unnamed: 7
    1      1    X28+  2003/11/04     486  19:29   19:53  20:06  MovieView archive
    2      2    X20+  2001/04/02    9393  21:32   21:51  22:03  MovieView archive
    3      3  X17.2+  2003/10/28     486  09:51   11:10  11:24  MovieView archive
    4      4    X17+  2005/09/07     808  17:17   17:40  18:03  MovieView archive
    5      5   X14.4  2001/04/15    9415  13:19   13:50  13:55  MovieView archive
    6      6     X10  2003/10/29     486  20:37   20:49  21:01  MovieView archive
    7      7    X9.4  1997/11/06    8100  11:49   11:55  12:01  MovieView archive
    8      8    X9.3  2017/09/06    2673  11:53   12:02  12:10  MovieView archive
    9      9      X9  2006/12/05     930  10:18   10:35  10:45  MovieView archive
    10    10    X8.3  2003/11/02     486  17:03   17:25  17:39  MovieView archive
    11    11    X8.2  2017/09/10    2673  15:35   16:06  16:31  MovieView archive
    12    12    X7.1  2005/01/20     720  06:36   07:01  07:26  MovieView archive
    13    13    X6.9  2011/08/09    1263  07:48   08:05  08:08  MovieView archive
    14    14    X6.5  2006/12/06     930  18:29   18:47  19:00  MovieView archive
    15    15    X6.2  2005/09/09     808  19:13   20:04  20:36  MovieView archive
    16    16    X6.2  2001/12/13    9733  14:20   14:30  14:35  MovieView archive
    17    17    X5.7  2000/07/14    9077  10:03   10:24  10:43  MovieView archive
    18    18    X5.6  2001/04/06    9415  19:10   19:21  19:31  MovieView archive
    19    19    X5.4  2012/03/07    1429  00:02   00:24  00:40  MovieView archive
    20    20    X5.4  2005/09/08     808  20:52   21:06  21:17  MovieView archive
    21    21    X5.4  2003/10/23     486  08:19   08:35  08:49  MovieView archive
    22    22    X5.3  2001/08/25    9591  16:23   16:45  17:04  MovieView archive
    23    23    X4.9  2014/02/25    1990  00:39   00:49  01:03  MovieView archive
    24    24    X4.9  1998/08/18    8307  22:10   22:19  22:28       View archive
    25    25    X4.8  2002/07/23      39  00:18   00:35  00:47  MovieView archive
    26    26      X4  2000/11/26    9236  16:34   16:48  16:56  MovieView archive
    27    27    X3.9  2003/11/03     488  09:43   09:55  10:19  MovieView archive
    28    28    X3.9  1998/08/19    8307  21:35   21:45  21:50       View archive
    29    29    X3.8  2005/01/17     720  06:59   09:52  10:07  MovieView archive
    30    30    X3.7  1998/11/22    8384  06:30   06:42  06:49  MovieView archive
    31    31    X3.6  2005/09/09     808  09:42   09:59  10:08  MovieView archive
    32    32    X3.6  2004/07/16     649  13:49   13:55  14:01  MovieView archive
    33    33    X3.6  2003/05/28     365  00:17   00:27  00:39  MovieView archive
    34    34    X3.4  2006/12/13     930  02:14   02:40  02:57  MovieView archive
    35    35    X3.4  2001/12/28    9767  20:02   20:45  21:32  MovieView archive
    36    36    X3.3  2013/11/05    1890  22:07   22:12  22:15  MovieView archive
    37    37    X3.3  2002/07/20      39  21:04   21:30  21:54  MovieView archive
    38    38    X3.3  1998/11/28    8395  04:54   05:52  06:13  MovieView archive
    39    39    X3.2  2013/05/14    1748  00:00   01:11  01:20  MovieView archive
    40    40    X3.1  2014/10/24    2192  21:07   21:41  22:13  MovieView archive
    41    41    X3.1  2002/08/24      69  00:49   01:12  01:31  MovieView archive
    42    42      X3  2002/07/15      30  19:59   20:08  20:14  MovieView archive
```

```
43    43    X2.8  2013/05/13    1748  15:48   16:05  16:16  MovieView archive
44    44    X2.8  2001/12/11    9733  07:58   08:08  08:14  MovieView archive
45    45    X2.8  1998/08/18    8307  08:14   08:24  08:32       View archive
46    46    X2.7  2015/05/05    2339  22:05   22:11  22:15  MovieView archive
47    47    X2.7  2003/11/03     488  01:09   01:30  01:45  MovieView archive
48    48    X2.7  1998/05/06    8210  07:58   08:09  08:20  MovieView archive
49    49    X2.6  2005/01/15     720  22:25   23:02  23:31  MovieView archive
50    50    X2.6  2001/09/24    9632  09:32   10:38  11:09  MovieView archive
```

[3]:

```python
# This block is for part 1 step 2

using_table=using_table.drop(axis=1,labels='Unnamed: 7')

using_table['Start'] = pd.to_datetime(using_table['date'].apply(str)+'␣
 ↪'+using_table['Start'])

using_table['Maximum'] = pd.to_datetime(using_table['date'].apply(str)+'␣
 ↪'+using_table['Maximum'])

using_table['End'] = pd.to_datetime(using_table['date'].apply(str)+'␣
 ↪'+using_table['End'])

using_table['date']=pd.to_datetime(using_table['date'])

using_table['Region']=using_table['Region'].replace(to_replace='-',value=np.NaN)

def clear_x_class(x):

    if x[len(x)-1]=='+' or x[len(x)-1]=='.':
        return x[0:len(x)-1]
    else:
        return x


answer_for_step2=using_table.copy()



answer_for_step2['x_class']=answer_for_step2['x_class'].apply(clear_x_class)

answer_for_step2





# Here is the answer for Part 1 Step 2
# I use pd.to_datetime() and apply() to convert the start, max and end column␣
 ↪into datestamp.
# Also, I clean the x_class column so that any entry end with '+' or '-' will␣
 ↪be replaced with
# the same data just without the last '+' or '-'

# Then, I use replace() to replace the '-' to np.NaN
```

```
[3]:     rank x_class       date  Region               Start             Maximum  \
    1       1     X28 2003-11-04     486 2003-11-04 19:29:00 2003-11-04 19:53:00
    2       2     X20 2001-04-02    9393 2001-04-02 21:32:00 2001-04-02 21:51:00
    3       3   X17.2 2003-10-28     486 2003-10-28 09:51:00 2003-10-28 11:10:00
    4       4     X17 2005-09-07     808 2005-09-07 17:17:00 2005-09-07 17:40:00
    5       5   X14.4 2001-04-15    9415 2001-04-15 13:19:00 2001-04-15 13:50:00
    6       6     X10 2003-10-29     486 2003-10-29 20:37:00 2003-10-29 20:49:00
    7       7    X9.4 1997-11-06    8100 1997-11-06 11:49:00 1997-11-06 11:55:00
    8       8    X9.3 2017-09-06    2673 2017-09-06 11:53:00 2017-09-06 12:02:00
    9       9      X9 2006-12-05     930 2006-12-05 10:18:00 2006-12-05 10:35:00
    10     10    X8.3 2003-11-02     486 2003-11-02 17:03:00 2003-11-02 17:25:00
    11     11    X8.2 2017-09-10    2673 2017-09-10 15:35:00 2017-09-10 16:06:00
    12     12    X7.1 2005-01-20     720 2005-01-20 06:36:00 2005-01-20 07:01:00
    13     13    X6.9 2011-08-09    1263 2011-08-09 07:48:00 2011-08-09 08:05:00
    14     14    X6.5 2006-12-06     930 2006-12-06 18:29:00 2006-12-06 18:47:00
    15     15    X6.2 2005-09-09     808 2005-09-09 19:13:00 2005-09-09 20:04:00
    16     16    X6.2 2001-12-13    9733 2001-12-13 14:20:00 2001-12-13 14:30:00
    17     17    X5.7 2000-07-14    9077 2000-07-14 10:03:00 2000-07-14 10:24:00
    18     18    X5.6 2001-04-06    9415 2001-04-06 19:10:00 2001-04-06 19:21:00
    19     19    X5.4 2012-03-07    1429 2012-03-07 00:02:00 2012-03-07 00:24:00
    20     20    X5.4 2005-09-08     808 2005-09-08 20:52:00 2005-09-08 21:06:00
    21     21    X5.4 2003-10-23     486 2003-10-23 08:19:00 2003-10-23 08:35:00
    22     22    X5.3 2001-08-25    9591 2001-08-25 16:23:00 2001-08-25 16:45:00
    23     23    X4.9 2014-02-25    1990 2014-02-25 00:39:00 2014-02-25 00:49:00
    24     24    X4.9 1998-08-18    8307 1998-08-18 22:10:00 1998-08-18 22:19:00
    25     25    X4.8 2002-07-23      39 2002-07-23 00:18:00 2002-07-23 00:35:00
    26     26      X4 2000-11-26    9236 2000-11-26 16:34:00 2000-11-26 16:48:00
    27     27    X3.9 2003-11-03     488 2003-11-03 09:43:00 2003-11-03 09:55:00
    28     28    X3.9 1998-08-19    8307 1998-08-19 21:35:00 1998-08-19 21:45:00
    29     29    X3.8 2005-01-17     720 2005-01-17 06:59:00 2005-01-17 09:52:00
    30     30    X3.7 1998-11-22    8384 1998-11-22 06:30:00 1998-11-22 06:42:00
    31     31    X3.6 2005-09-09     808 2005-09-09 09:42:00 2005-09-09 09:59:00
    32     32    X3.6 2004-07-16     649 2004-07-16 13:49:00 2004-07-16 13:55:00
    33     33    X3.6 2003-05-28     365 2003-05-28 00:17:00 2003-05-28 00:27:00
    34     34    X3.4 2006-12-13     930 2006-12-13 02:14:00 2006-12-13 02:40:00
    35     35    X3.4 2001-12-28    9767 2001-12-28 20:02:00 2001-12-28 20:45:00
    36     36    X3.3 2013-11-05    1890 2013-11-05 22:07:00 2013-11-05 22:12:00
    37     37    X3.3 2002-07-20      39 2002-07-20 21:04:00 2002-07-20 21:30:00
    38     38    X3.3 1998-11-28    8395 1998-11-28 04:54:00 1998-11-28 05:52:00
```

```
39    39    X3.2 2013-05-14    1748 2013-05-14 00:00:00 2013-05-14 01:11:00
40    40    X3.1 2014-10-24    2192 2014-10-24 21:07:00 2014-10-24 21:41:00
41    41    X3.1 2002-08-24      69 2002-08-24 00:49:00 2002-08-24 01:12:00
42    42      X3 2002-07-15      30 2002-07-15 19:59:00 2002-07-15 20:08:00
43    43    X2.8 2013-05-13    1748 2013-05-13 15:48:00 2013-05-13 16:05:00
44    44    X2.8 2001-12-11    9733 2001-12-11 07:58:00 2001-12-11 08:08:00
45    45    X2.8 1998-08-18    8307 1998-08-18 08:14:00 1998-08-18 08:24:00
46    46    X2.7 2015-05-05    2339 2015-05-05 22:05:00 2015-05-05 22:11:00
47    47    X2.7 2003-11-03     488 2003-11-03 01:09:00 2003-11-03 01:30:00
48    48    X2.7 1998-05-06    8210 1998-05-06 07:58:00 1998-05-06 08:09:00
49    49    X2.6 2005-01-15     720 2005-01-15 22:25:00 2005-01-15 23:02:00
50    50    X2.6 2001-09-24    9632 2001-09-24 09:32:00 2001-09-24 10:38:00

                   End
1  2003-11-04 20:06:00
2  2001-04-02 22:03:00
3  2003-10-28 11:24:00
4  2005-09-07 18:03:00
5  2001-04-15 13:55:00
6  2003-10-29 21:01:00
7  1997-11-06 12:01:00
8  2017-09-06 12:10:00
9  2006-12-05 10:45:00
10 2003-11-02 17:39:00
11 2017-09-10 16:31:00
12 2005-01-20 07:26:00
13 2011-08-09 08:08:00
14 2006-12-06 19:00:00
15 2005-09-09 20:36:00
16 2001-12-13 14:35:00
17 2000-07-14 10:43:00
18 2001-04-06 19:31:00
19 2012-03-07 00:40:00
20 2005-09-08 21:17:00
21 2003-10-23 08:49:00
22 2001-08-25 17:04:00
23 2014-02-25 01:03:00
24 1998-08-18 22:28:00
25 2002-07-23 00:47:00
26 2000-11-26 16:56:00
27 2003-11-03 10:19:00
28 1998-08-19 21:50:00
29 2005-01-17 10:07:00
30 1998-11-22 06:49:00
31 2005-09-09 10:08:00
32 2004-07-16 14:01:00
33 2003-05-28 00:39:00
```

```
34 2006-12-13 02:57:00
35 2001-12-28 21:32:00
36 2013-11-05 22:15:00
37 2002-07-20 21:54:00
38 1998-11-28 06:13:00
39 2013-05-14 01:20:00
40 2014-10-24 22:13:00
41 2002-08-24 01:31:00
42 2002-07-15 20:14:00
43 2013-05-13 16:16:00
44 2001-12-11 08:14:00
45 1998-08-18 08:32:00
46 2015-05-05 22:15:00
47 2003-11-03 01:45:00
48 1998-05-06 08:20:00
49 2005-01-15 23:31:00
50 2001-09-24 11:09:00
```

[4]:

```
# This block is for part 1 step 3

r= requests.get("https://cdaw.gsfc.nasa.gov/CME_list/radio/waves_type2.html")

web_text=r.text

text_arr=web_text.split("\n")

text_arr[15]

new_arr = text_arr[15:]

new_arr=new_arr[:518]

def cleanhtml(raw_html):
  cleanr = re.compile('<.*?>')
  cleantext = re.sub(cleanr, '', raw_html)
  return cleantext

l=[]

for s in new_arr:
    l.append(cleanhtml(s))

for i in range(0,len(l)):
   l[i]=l[i].split()


for i in range(0,len(l)):
    l[i]=l[i][:14]




df = pd.DataFrame(l)



df[0]=df[0]+" "+df[1]

df=df.drop(1, axis=1)

df[0]=pd.to_datetime(df[0])

df[2]=df[2]+" "+df[3]

copy=[]
for i in range(0,len(df)):
    copy.append(str(df[0][i].year))

df['year']=copy


def my_to_datetime(date_str):
    if date_str[11:13] == '--':
        return np.nan

    if date_str[11:13] != '24':
        return pd.to_datetime(date_str, format='%Y/%m/%d %H:%M')


    date_str = date_str[0:11] + '00' + date_str[13:]
    return pd.to_datetime(date_str, format='%Y/%m/%d %H:%M') + \
           datetime.timedelta(days=1)


df[2]=df['year']+"/"+df[2]




df[2] = df[2].apply(my_to_datetime)

df[9] = df['year']+"/"+df[9]+" "+df[10]
df[9]=df[9].apply(my_to_datetime)
```

```
df=df.drop(3, axis=1)

df=df.drop('year',axis=1)
df=df.drop(10,axis=1)


df = df.rename(columns = {0: 'start_datetime',2:  'end_datetime',4:␣
 ↪'start_frequency',5: 'end_frequency',6: 'flare_location',7: 'flare_region',8:
 ↪ 'importance',9:  'cme_datetime',11:  'cpa',12: 'width',13: 'speed'})

df

# Here is answer for Part 1 Step 3

# This is similar to Part 1 Step 1. First, use requests to get the data from␣
 ↪the website.
# Since this table is stored in a long array, I use split to make a list and␣
 ↪then store
# it into a dataframe.

# Then, use the same trick on the start_datetime, end_datetime and cme_datetime␣
 ↪(by apply())
# However, since there are 24:00 in some column, I need to create my␣
 ↪my_to_datetime().

# Note that the merge the date and time into datetime should be a part of the␣
 ↪next step, but I did it here.

# At last, I drop the column I used for cleaning the data and rename the␣
 ↪columns.
```

[4]:

```
         start_datetime        end_datetime start_frequency end_frequency  \
0   1997-04-01 14:00:00 1997-04-01 14:15:00            8000          4000
1   1997-04-07 14:30:00 1997-04-07 17:30:00           11000          1000
2   1997-05-12 05:15:00 1997-05-14 16:00:00           12000            80
3   1997-05-21 20:20:00 1997-05-21 22:00:00            5000           500
4   1997-09-23 21:53:00 1997-09-23 22:16:00            6000          2000
..                  …                   …             …             …
513 2017-09-04 20:27:00 2017-09-05 04:54:00           14000           210
514 2017-09-06 12:05:00 2017-09-07 08:00:00           16000            70
515 2017-09-10 16:02:00 2017-09-11 06:50:00           16000           150
516 2017-09-12 07:38:00 2017-09-12 07:43:00           16000         13000
517 2017-09-17 11:45:00 2017-09-17 12:35:00           16000           900

    flare_location flare_region importance        cme_datetime   cpa width  \
0           S25E16         8026       M1.3 1997-04-01 15:18:00    74    79
```

```
1            S28E19         8027        C6.8 1997-04-07 14:27:00  Halo   360
2            N21W08         8038        C1.3 1997-05-12 05:30:00  Halo   360
3            N05W12         8040        M1.3 1997-05-21 21:00:00   263   165
4            S29E25         8088        C1.4 1997-09-23 22:02:00   133   155
..              …             …        …                   …    …    …
513          S10W12        12673        M5.5 2017-09-04 20:12:00  Halo   360
514          S08W33        12673        X9.3 2017-09-06 12:24:00  Halo   360
515          S09W92        -----        X8.3 2017-09-10 16:00:00  Halo   360
516          N08E48        12680        C3.0 2017-09-12 08:03:00   124    96
517         S08E170        -----        ---- 2017-09-17 12:00:00  Halo   360

    speed
0     312
1     878
2     464
3     296
4     712
..    …
513  1418
514  1571
515  3163
516   252
517  1385

[518 rows x 11 columns]
```

[5]:

```python
# This block is for part 1 step 4

is_halo_list=[]
cpa_list=[]
for i in range(0,len(df)):
    if df['cpa'][i]=="Halo":
        is_halo_list.append(True)
        cpa_list.append('NA')
    else:
        is_halo_list.append(False)
        cpa_list.append(df['cpa'][i])



is_halo_list
df['is_halo']=is_halo_list
df['cpa'] = cpa_list

width_lower_bound_list=[]
for i in range(0, len(df)):
    if ">" in df['width'][i]:
        width_lower_bound_list.append(True)
    else:
        width_lower_bound_list.append(False)

df['width_lower_bound']=width_lower_bound_list


answer_for_step_4=df.copy()
answer_for_step_4['flare_region']=answer_for_step_4['flare_region'].
 ↪replace(to_replace='-----',value=np.NaN)
answer_for_step_4['importance']=answer_for_step_4['importance'].
 ↪replace(to_replace='----',value=np.NaN)
answer_for_step_4['cpa']=answer_for_step_4['cpa'].
 ↪replace(to_replace='----',value=np.NaN)
answer_for_step_4['width']=answer_for_step_4['width'].
 ↪replace(to_replace='----',value=np.NaN)
answer_for_step_4['speed']=answer_for_step_4['speed'].
 ↪replace(to_replace='----',value=np.NaN)




answer_for_step_4
# Answer for Part 1 Step 4

# First, I create a list that is used to store new values for the cpa column,␣
 ↪and replace
# the cpa column with the list. At the same time, I have a is_halo list to␣
 ↪store whether
# a row is halo or not, and put the list into the dataframe as the is_halo␣
 ↪column.

# I do the similar thing for the width_lower_bound column to create this new␣
 ↪column. First,
# create a list with appropriate data and put the list into the dataframe.

# Then, I notice that only flare_region, importance, cpa, width and speed␣
 ↪column has missing
# value, and I replace them by np.NaN

# Since I change the datetime column in the last step, I do not need to do it␣
 ↪in this step.
```

[5]:

```
        start_datetime        end_datetime start_frequency end_frequency  \
0  1997-04-01 14:00:00 1997-04-01 14:15:00            8000          4000
1  1997-04-07 14:30:00 1997-04-07 17:30:00           11000          1000
2  1997-05-12 05:15:00 1997-05-14 16:00:00           12000            80
```

```
3   1997-05-21 20:20:00 1997-05-21 22:00:00            5000           500
4   1997-09-23 21:53:00 1997-09-23 22:16:00            6000          2000
..                  …                 …             …           …
513 2017-09-04 20:27:00 2017-09-05 04:54:00           14000           210
514 2017-09-06 12:05:00 2017-09-07 08:00:00           16000            70
515 2017-09-10 16:02:00 2017-09-11 06:50:00           16000           150
516 2017-09-12 07:38:00 2017-09-12 07:43:00           16000         13000
517 2017-09-17 11:45:00 2017-09-17 12:35:00           16000           900

    flare_location flare_region importance         cme_datetime  cpa width  \
0           S25E16         8026       M1.3 1997-04-01 15:18:00   74    79
1           S28E19         8027       C6.8 1997-04-07 14:27:00   NA   360
2           N21W08         8038       C1.3 1997-05-12 05:30:00   NA   360
3           N05W12         8040       M1.3 1997-05-21 21:00:00  263   165
4           S29E25         8088       C1.4 1997-09-23 22:02:00  133   155
..               …           …        …                  …   …    …
513         S10W12        12673       M5.5 2017-09-04 20:12:00   NA   360
514         S08W33        12673       X9.3 2017-09-06 12:24:00   NA   360
515         S09W92          NaN       X8.3 2017-09-10 16:00:00   NA   360
516         N08E48        12680       C3.0 2017-09-12 08:03:00  124    96
517        S08E170          NaN        NaN 2017-09-17 12:00:00   NA   360

    speed  is_halo  width_lower_bound
0     312    False              False
1     878     True              False
2     464     True              False
3     296    False              False
4     712    False              False
..     …      …                …
513  1418     True              False
514  1571     True              False
515  3163     True              False
516   252    False              False
517  1385     True              False

[518 rows x 13 columns]
```

[6]:

```python
# This block is for part 2 question 1
df2 = pd.DataFrame()


for i in range(0,len(df)):
    if df['importance'][i] is np.nan:
        continue

    if 'X' in df['importance'][i]:
        df2=df2.append(df.loc[[i]])
```

```
#df2['just_date']=df2['start_datetime'].apply(lambda x:str(x)[0:10])



df2.index=range(92)

importance_list = []
for i in range(0,len(df2)):
    importance_list.append( float(df2['importance'][i][1:]))

df2['real_importance']=importance_list

df2.sort_values(by=['real_importance'],ascending=False,inplace=True)

top_50=df2.head(n=50)

top_50.index=range(1,51)

top_50=top_50.drop('real_importance',axis=1)

top_50['flare_region']=top_50['flare_region'].
 ↪replace(to_replace='-----',value=np.NaN)
top_50['importance']=top_50['importance'].replace(to_replace='----',value=np.
 ↪NaN)
top_50['cpa']=top_50['cpa'].replace(to_replace='----',value=np.NaN)
top_50['width']=top_50['width'].replace(to_replace='----',value=np.NaN)
top_50['speed']=top_50['speed'].replace(to_replace='----',value=np.NaN)


top_50
# Comparing the top 50 solar flare table from NASA to the one from SWL,
# we can find that SWL has some data that NASA does not have. For example,
# the 4th solar flare in SWL table, on 2005/09/07 does not have have a
# corresponding row in the NASA table.

# Therefore, my conclusion is that SWL cannot be replicated well from
# NASA, since the NASA data miss at least 15 rows that SWL has, which is
# nearly 30%.

# Here is the answer for Part2 Question 1

# Some explanation for this question:
# df2 is a dataframe with the rows from NASA that have 'X' in the importance␣
 ↪column.
```

```
# top_50 is a dataframe with the highest 50 importance in df2, with descending␣
↪order.
```

```
[6]:        start_datetime        end_datetime start_frequency end_frequency  \
     1  2003-11-04 20:00:00 2003-11-05 00:00:00           10000           200
     2  2001-04-02 22:05:00 2001-04-03 02:30:00           14000           250
     3  2003-10-28 11:10:00 2003-10-30 00:00:00           14000            40
     4  2001-04-15 14:05:00 2001-04-16 13:00:00           14000            40
     5  2003-10-29 20:55:00 2003-10-30 00:00:00           11000           500
     6  1997-11-06 12:20:00 1997-11-07 08:30:00           14000           100
     7  2017-09-06 12:05:00 2017-09-07 08:00:00           16000            70
     8  2006-12-05 10:50:00 2006-12-05 20:00:00           14000           250
     9  2003-11-02 17:30:00 2003-11-03 01:00:00           12000           250
     10 2017-09-10 16:02:00 2017-09-11 06:50:00           16000           150
     11 2005-01-20 07:15:00 2005-01-20 16:30:00           14000            25
     12 2011-08-09 08:20:00 2011-08-09 08:35:00           16000          4000
     13 2006-12-06 19:00:00 2006-12-09 00:00:00           16000            30
     14 2005-09-09 19:45:00 2005-09-09 22:00:00           10000            50
     15 2000-07-14 10:30:00 2000-07-15 14:30:00           14000            80
     16 2001-04-06 19:35:00 2001-04-07 01:50:00           14000           230
     17 2012-03-07 01:00:00 2012-03-08 19:00:00           16000            30
     18 2001-08-25 16:50:00 2001-08-25 23:00:00            8000           170
     19 2014-02-25 00:56:00 2014-02-25 11:28:00           14000           100
     20 2002-07-23 00:50:00 2002-07-23 04:00:00           11000           400
     21 2000-11-26 17:00:00 2000-11-26 17:15:00           14000          7000
     22 2003-11-03 10:00:00 2003-11-03 12:30:00            6000           400
     23 2005-01-17 10:00:00 2005-01-17 10:35:00            6100          1500
     24 2003-05-28 01:00:00 2003-05-29 00:30:00            1000           200
     25 2006-12-13 02:45:00 2006-12-13 10:40:00           12000           150
     26 2001-12-28 20:35:00 2001-12-29 03:00:00           14000           350
     27 2002-07-20 21:30:00 2002-07-20 22:20:00           10000          2000
     28 2013-05-14 01:16:00 2013-05-14 08:20:00           16000           240
     29 2002-08-24 01:45:00 2002-08-24 03:25:00            5000           400
     30 2013-05-13 16:15:00 2013-05-13 19:10:00           16000           300
     31 2015-05-05 22:24:00 2015-05-05 23:14:00           14000           500
     32 1998-05-06 08:25:00 1998-05-06 08:35:00           14000          5000
     33 2003-11-03 01:15:00 2003-11-03 01:25:00            3000          1500
     34 2005-01-15 23:00:00 2005-01-17 00:00:00            3000            40
     35 2001-09-24 10:45:00 2001-09-25 20:00:00            7000            30
     36 1997-11-27 13:30:00 1997-11-27 14:00:00           14000          7000
     37 2004-11-10 02:25:00 2004-11-10 03:40:00           14000          1000
     38 2001-04-10 05:24:00 2001-04-11 00:00:00           14000           100
     39 2000-11-24 15:25:00 2000-11-24 22:00:00           14000           200
     40 2000-06-06 15:20:00 2000-06-08 09:00:00           14000            40
     41 2011-02-15 02:10:00 2011-02-15 07:00:00           16000           400
     42 2005-09-10 21:45:00 2005-09-11 01:00:00           14000           200
     43 2011-09-06 22:30:00 2011-09-07 15:40:00           16000           150
```

```
44 2013-10-25 15:08:00 2013-10-25 22:32:00           16000           200
45 1997-11-04 06:00:00 1997-11-05 04:30:00           14000           100
46 2000-11-24 05:10:00 2000-11-24 15:00:00           14000           100
47 2001-04-12 10:20:00 2001-04-12 10:40:00           14000          7000
48 2004-11-07 16:25:00 2004-11-08 20:00:00           14000            60
49 2005-01-17 09:25:00 2005-01-17 16:00:00           14000            30
50 2000-11-25 19:00:00 2000-11-25 19:35:00            6000          2000

   flare_location flare_region importance        cme_datetime  cpa width  \
1          S19W83        10486       X28. 2003-11-04 19:54:00   NA   360
2          N19W72         9393       X20. 2001-04-02 22:06:00  261   244
3          S16E08        10486       X17. 2003-10-28 11:30:00   NA   360
4          S20W85         9415       X14. 2001-04-15 14:06:00  245   167
5          S15W02        10486       X10. 2003-10-29 20:54:00   NA   360
6          S18W63         8100       X9.4 1997-11-06 12:10:00   NA   360
7          S08W33        12673       X9.3 2017-09-06 12:24:00   NA   360
8          S07E68        10930       X9.0                 NaT  NaN   NaN
9          S14W56        10486       X8.3 2003-11-02 17:30:00   NA   360
10         S09W92          NaN       X8.3 2017-09-10 16:00:00   NA   360
11         N14W61        10720       X7.1 2005-01-20 06:54:00   NA   360
12         N17W69        11263       X6.9 2011-08-09 08:12:00   NA   360
13         S05E64        10930       X6.5                 NaT  NaN   NaN
14         S12E67        10808       X6.2 2005-09-09 19:48:00   NA   360
15         N22W07         9077       X5.7 2000-07-14 10:54:00   NA   360
16         S21E31         9415       X5.6 2001-04-06 19:30:00   NA   360
17         N17E27        11429       X5.4 2012-03-07 00:24:00   NA   360
18         S17E34         9591       X5.3 2001-08-25 16:50:00   NA   360
19         S12E82        11990       X4.9 2014-02-25 01:25:00   NA   360
20         S13E72        10039       X4.8 2002-07-23 00:42:00   NA   360
21         N18W38         9236       X4.0 2000-11-26 17:06:00   NA   360
22         N08W77        10488       X3.9 2003-11-03 10:06:00  293   103
23         N15W25        10720       X3.8 2005-01-17 09:54:00   NA   360
24         S07W20        10365       X3.6 2003-05-28 00:50:00   NA   360
25         S06W23        10930       X3.4 2006-12-13 02:54:00   NA   360
26         S26E90         9756       X3.4 2001-12-28 20:30:00   NA   360
27         S13E90        10039       X3.3 2002-07-20 22:06:00   NA   360
28         N08E77        11748       X3.2 2013-05-14 01:25:00   NA   360
29         S02W81        10069       X3.1 2002-08-24 01:27:00   NA   360
30         N11E85        11748       X2.8 2013-05-13 16:07:00   NA   360
31         N15E79        12339       X2.7 2015-05-05 22:24:00   NA   360
32         S11W65         8210       X2.7 1998-05-06 08:29:00  309   190
33         N10W83        10488       X2.7 2003-11-03 01:59:00  304    65
34         N15W05        10720       X2.6 2005-01-15 23:06:00   NA   360
35         S16E23         9632       X2.6 2001-09-24 10:30:00   NA   360
36         N17E63         8113       X2.6 1997-11-27 13:56:00   98    91
37         N09W49        10696       X2.5 2004-11-10 02:26:00   NA   360
38         S23W09         9415       X2.3 2001-04-10 05:30:00   NA   360
```

```
39         N22W07          9236       X2.3 2000-11-24 15:30:00   NA   360
40         N20E18          9026       X2.3 2000-06-06 15:54:00   NA   360
41         S20W12         11158       X2.2 2011-02-15 02:24:00   NA   360
42         S13E47         10808       X2.1 2005-09-10 21:52:00   NA   360
43         N14W18         11283       X2.1 2011-09-06 23:05:00   NA   360
44         S06E69         11882       X2.1 2013-10-25 15:12:00   NA   360
45         S14W33          8100       X2.1 1997-11-04 06:10:00   NA   360
46         N20W05          9236       X2.0 2000-11-24 05:30:00   NA   360
47         S19W43          9415       X2.0 2001-04-12 10:31:00   NA   360
48         N09W17         10696       X2.0 2004-11-07 16:54:00   NA   360
49         N15W25         10720       X2.0 2005-01-17 09:30:00   NA   360
50         N20W23          9236       X1.9 2000-11-25 19:31:00   NA   360

   speed  is_halo  width_lower_bound
1   2657     True              False
2   2505    False              False
3   2459     True              False
4   1199    False              False
5   2029     True              False
6   1556     True              False
7   1571     True              False
8    NaN    False              False
9   2598     True              False
10  3163     True              False
11   882     True              False
12  1610     True              False
13   NaN    False              False
14  2257     True              False
15  1674     True              False
16  1270     True              False
17  2684     True              False
18  1433     True              False
19  2147     True              False
20  2285     True              False
21   980     True              False
22  1420    False              False
23  2547     True              False
24  1366     True              False
25  1774     True              False
26  2216     True              False
27  1941     True              False
28  2625     True              False
29  1913     True              False
30  1850     True              False
31   715     True              False
32  1099    False              False
33   827    False              False
```

```
34  2861     True               False
35  2402     True               False
36   441    False               False
37  3387     True               False
38  2411     True               False
39  1245     True               False
40  1119     True               False
41   669     True               False
42  1893     True               False
43   575     True               False
44  1081     True               False
45   785     True               False
46  1289     True               False
47  1184     True               False
48  1759     True               False
49  2094     True               False
50   671     True               False
```

[7]:

```
# This block is for part 2 question 2

SWL_table = using_table
NASA_table = df
NASA_table['best_matched_rank']=pd.Series(np.nan, index=NASA_table.index)
def clear_x_class(x):

    if x[len(x)-1]=='+' or x[len(x)-1]=='.':
        return x[0:len(x)-1]
    else:
        return x


SWL_table['x_class']=SWL_table['x_class'].apply(clear_x_class)
#NASA_table['importance']=NASA_table['importance'].apply(clear_x_class)
a=NASA_table['importance'][514]
SWL_table['date']=SWL_table['date'].apply(lambda x: str(x)[0:10])
#print(SWL_table['date'][1])
NASA_table['approx_date']=NASA_table['start_datetime'].apply(lambda x: str(x)[0:
 ↪10])


def best_matched_rank(index):
    rl=[]
    for i, row in SWL_table.iterrows():
        if row['x_class']== NASA_table['importance'][index] and␣
 ↪row['date']==NASA_table['approx_date'][index]:

            rl.append(row['rank'])
            #print("nothing")
    return rl



for i, row in NASA_table.iterrows():

    temp = best_matched_rank(i)
    if(temp != []):
        if(len(temp)!=1):
            print(temp)
        NASA_table['best_matched_rank'][i]=temp[0]
        #count=count+1
        #print(count)

NASA_table.drop(columns='approx_date',axis=1)

NASA_table['flare_region']=NASA_table['flare_region'].
 ↪replace(to_replace='-----',value=np.NaN)
NASA_table['importance']=NASA_table['importance'].
 ↪replace(to_replace='----',value=np.NaN)
NASA_table['cpa']=NASA_table['cpa'].replace(to_replace='----',value=np.NaN)
NASA_table['width']=NASA_table['width'].replace(to_replace='----',value=np.NaN)
NASA_table['speed']=NASA_table['speed'].replace(to_replace='----',value=np.NaN)
NASA_table

# I define a SWL entry match best matches a NASA entry if and only if they␣
 ↪happen on the same date (approximate to day)
# and have the same x_class
# For my best match, there is no case that a NASA entry has more than one SWL␣
 ↪matches.

# Here is the answer for Part 2 Question 2
```

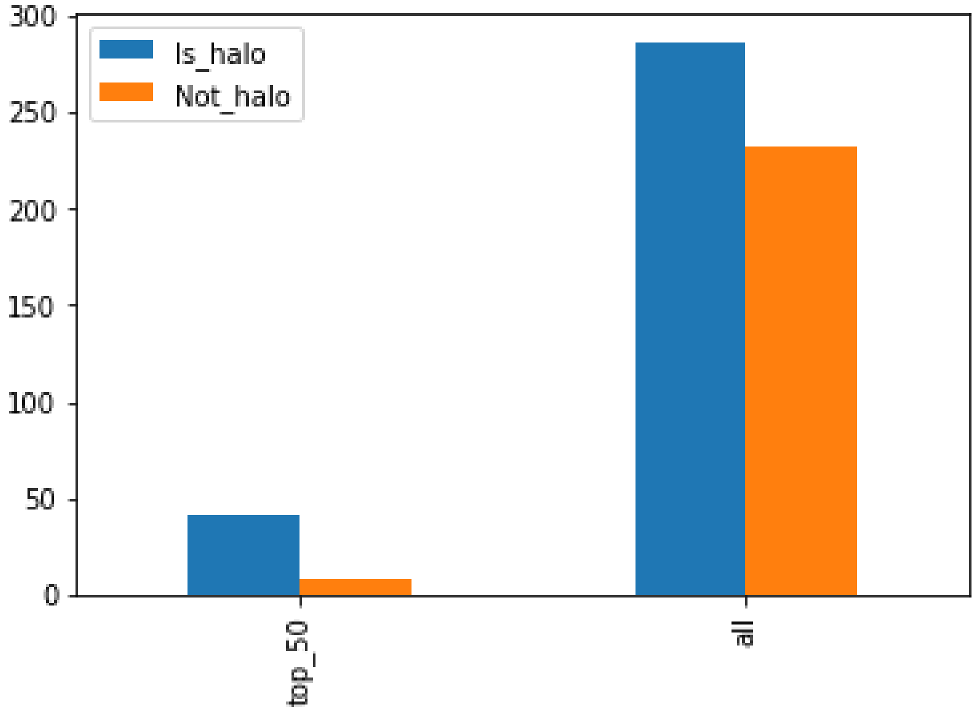
```
<ipython-input-7-af21fc9b78e8>:40: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  NASA_table['best_matched_rank'][i]=temp[0]
```

```
[7]:          start_datetime        end_datetime start_frequency end_frequency  \
     0   1997-04-01 14:00:00 1997-04-01 14:15:00            8000          4000
     1   1997-04-07 14:30:00 1997-04-07 17:30:00           11000          1000
     2   1997-05-12 05:15:00 1997-05-14 16:00:00           12000            80
     3   1997-05-21 20:20:00 1997-05-21 22:00:00            5000           500
```

```
4   1997-09-23 21:53:00 1997-09-23 22:16:00            6000           2000
..                  …                 …             …           …
513 2017-09-04 20:27:00 2017-09-05 04:54:00           14000            210
514 2017-09-06 12:05:00 2017-09-07 08:00:00           16000             70
515 2017-09-10 16:02:00 2017-09-11 06:50:00           16000            150
516 2017-09-12 07:38:00 2017-09-12 07:43:00           16000          13000
517 2017-09-17 11:45:00 2017-09-17 12:35:00           16000            900

    flare_location flare_region importance        cme_datetime  cpa width  \
0           S25E16         8026       M1.3 1997-04-01 15:18:00   74    79
1           S28E19         8027       C6.8 1997-04-07 14:27:00   NA   360
2           N21W08         8038       C1.3 1997-05-12 05:30:00   NA   360
3           N05W12         8040       M1.3 1997-05-21 21:00:00  263   165
4           S29E25         8088       C1.4 1997-09-23 22:02:00  133   155
..             …            …         …                  …   …    …
513         S10W12        12673       M5.5 2017-09-04 20:12:00   NA   360
514         S08W33        12673       X9.3 2017-09-06 12:24:00   NA   360
515         S09W92          NaN       X8.3 2017-09-10 16:00:00   NA   360
516         N08E48        12680       C3.0 2017-09-12 08:03:00  124    96
517        S08E170          NaN        NaN 2017-09-17 12:00:00   NA   360

    speed  is_halo  width_lower_bound  best_matched_rank approx_date
0     312    False              False                NaN  1997-04-01
1     878     True              False                NaN  1997-04-07
2     464     True              False                NaN  1997-05-12
3     296    False              False                NaN  1997-05-21
4     712    False              False                NaN  1997-09-23
..    …      …                 …                 …         …
513  1418     True              False                NaN  2017-09-04
514  1571     True              False                8.0  2017-09-06
515  3163     True              False                NaN  2017-09-10
516   252    False              False                NaN  2017-09-12
517  1385     True              False                NaN  2017-09-17

[518 rows x 15 columns]
```

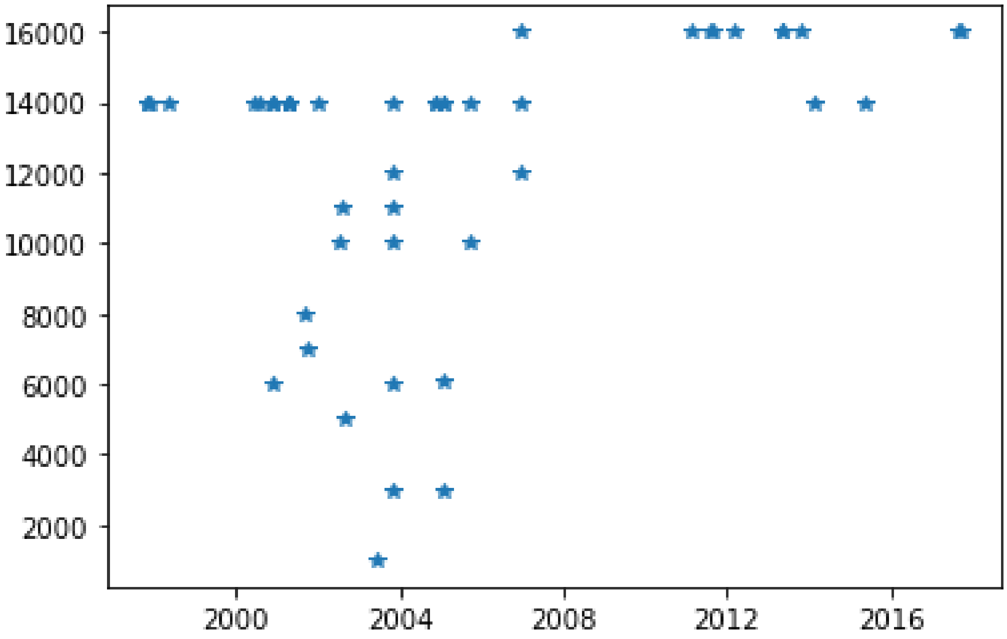
```
[8]: # This block is for part 2 question 3.

true_top_count = 0
true_all_count = 0
false_top_count = 0
false_all_count = 0

for i, r in top_50.iterrows():
    if r['is_halo']:
        true_top_count = true_top_count + 1
    else:
        false_top_count = false_top_count + 1

for i, r in NASA_table.iterrows():
    if r['is_halo']:
        true_all_count = true_all_count + 1
    else:
        false_all_count = false_all_count + 1

# print(true_top_count)
# print(true_all_count)

array_for_plot = np.array([ [true_top_count, false_top_count], [true_all_count␣
 ↪,false_all_count] ])

df_for_plot = pd.DataFrame(data=array_for_plot)

df_for_plot = df_for_plot.rename(columns = {0: 'Is_halo',1:'Not_halo'})

df_for_plot.index=['top_50','all']
df_for_plot

df_for_plot.plot.bar()


# Here is answer for Part2 Question 3

# I hope my plot can show the variation between the ratio of is_halo and␣
 ↪not_halo between the
# top 50 flares and all flares.

# The plot shows that the difference between the blue bar and the orange bar␣
 ↪comparing to
# their own height is more dramatic.
# Thus, the plot suggests that the ratio of is_halo flares and not_halo flares␣
 ↪in top_50 is much higher
# comparing to the ratio of all flares.

# As a conclusion, from the graph, I find there is strong correlation between␣
 ↪top 50 flares and whether it is
# a halo or not, because there are far more is_halo data in the top_50 flares␣
 ↪ratio is nearly 7:1, while for all data,
# the ratio is about 6:5. There exists a huge variation of this ratio in the␣
 ↪top_50 and all_data.

# Therefore, if a flare is in the top 50, the probability of it to be halo␣
 ↪would increase dramatically.
```

```
[8]: <AxesSubplot:>
```



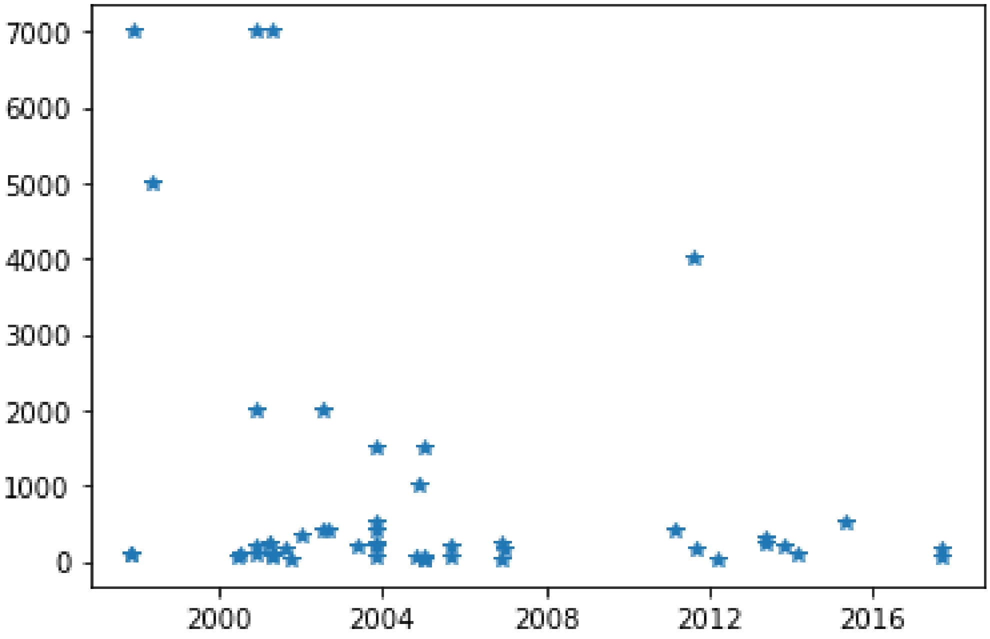
```
[9]: # The following is just a backup for Part2 Question 2.

     import matplotlib.pyplot as plt

     top_50['start_frequency']=top_50['start_frequency'].apply(lambda x:int(x))
     plt.plot(top_50['start_datetime'], top_50['start_frequency'],'*')
```

```
[9]: [<matplotlib.lines.Line2D at 0x7fed7d3e5be0>]
```



```
[10]: top_50['end_frequency']=top_50['end_frequency'].apply(lambda x:int(x))
      plt.plot(top_50['start_datetime'], top_50['end_frequency'],'*')
```

```
[10]: [<matplotlib.lines.Line2D at 0x7fed7d46d730>]
```

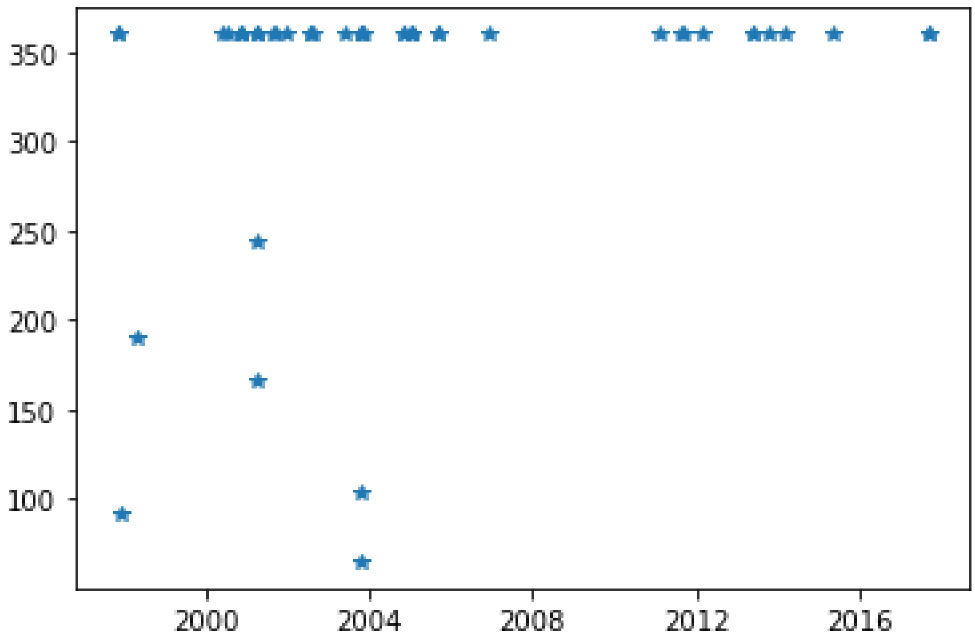


[11]:

```
top_50['width']=top_50['width'].replace(to_replace='----',value=np.NaN)
top_50.dropna(subset=['width'],inplace=True)
top_50['width']=top_50['width'].apply(lambda x:int(x))
plt.plot(top_50['start_datetime'], top_50['width'], '*')

# This is just a backup for part2 question 3.
# For the top 50 flares, the graphs show that there is a positive relation␣
 ↪between width and
# start frequency (i.e. If the flare has higher start frequency, it is more␣
 ↪likely to have larger width)
```

[11]:

```
[<matplotlib.lines.Line2D at 0x7fed7e7bbe50>]
```



[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

```
```

[ ]:

```
```