hw_4

December 7, 2022

1 Homework 4

1.1 Coding

Goal: Find complaint types that increased or decreased when COVID-19 hit New York Ciy: mid-March 2020.

1.1.1 Step 0: Setup

For this homework, instead of the data being provided, you will export it directly from the NYC Open Data Portal, as if you were working on your own project.

1. Download the data.
2. Visit the 311 data page.
3. From that page, filter the data to `Created Dates` between `01/01/2020 12:00:00 AM` and `03/31/2020 11:59:59 PM`.
4. It should say "Showing 311 Service Requests 1-100 out of 548,184" near the bottom of the screen. - It's ok if the total is slightly different.
5. Click `Export`.
6. Click `CSV`. It will start downloading a file.
7. Rename the file `311_covid.csv`.
8. Upload the CSV.
9. Read the data from `./<filename>.csv`.

- You may need to adjust the path, depending on where the CSV/notebook are.

If the above is taking a long time due to have a slow network connection or whatever else, load the data from:

https://storage.googleapis.com/python-public-policy/data/311_covid.csv.zip

1.1.2 Step 1: Load data

Read the data into a DataFrame called `df_2020`.

[1]:

```
import pandas as pd
import plotly.express as px
```

[2]:

```
# your code here
df_2020= pd.read_csv(
    "https://storage.googleapis.com/python-public-policy/data/311_covid.csv.zip"
)
```

```
/tmp/ipykernel_109/1941078713.py:2: DtypeWarning: Columns (34) have mixed types.
Specify dtype option on import or set low_memory=False.
  df_2020= pd.read_csv(
```

[3]:

```
df_2020.dtypes
```

[3]:

```
Unique Key                            int64
Created Date                         object
Closed Date                          object
Agency                               object
Agency Name                          object
Complaint Type                       object
Descriptor                           object
Location Type                        object
Incident Zip                        float64
Incident Address                     object
Street Name                          object
Cross Street 1                       object
Cross Street 2                       object
Intersection Street 1                object
Intersection Street 2                object
Address Type                         object
City                                 object
Landmark                             object
Facility Type                        object
Status                               object
Due Date                             object
Resolution Description               object
Resolution Action Updated Date       object
Community Board                      object
BBL                                 float64
Borough                              object
X Coordinate (State Plane)          float64
Y Coordinate (State Plane)          float64
Open Data Channel Type               object
Park Facility Name                   object
Park Borough                         object
Vehicle Type                         object
Taxi Company Borough                 object
Taxi Pick Up Location                object
Bridge Highway Name                  object
Bridge Highway Direction             object
Road Ramp                            object
Bridge Highway Segment               object
Latitude                            float64
```

```
Longitude                          float64
Location                            object
dtype: object
```

1.1.3 Step 2: Convert dates

Copy code from Lecture 4 to convert the `Created Date` to a `datetime`.

[4]:

```
# your code here
df_2020['Created Date']=pd.to_datetime(df_2020["Created Date"], format="%m/%d/
 ↪%Y %I:%M:%S %p")
df_2020["Closed Date"] = pd.to_datetime(df_2020["Closed Date"], format="%m/%d/
 ↪%Y %I:%M:%S %p")
df_2020.dtypes
```

[4]:

```
Unique Key                                   int64
Created Date                        datetime64[ns]
Closed Date                         datetime64[ns]
Agency                                      object
Agency Name                                 object
Complaint Type                              object
Descriptor                                  object
Location Type                               object
Incident Zip                               float64
Incident Address                            object
Street Name                                 object
Cross Street 1                              object
Cross Street 2                              object
Intersection Street 1                       object
Intersection Street 2                       object
Address Type                                object
City                                        object
Landmark                                    object
Facility Type                               object
Status                                      object
Due Date                                    object
Resolution Description                      object
Resolution Action Updated Date              object
Community Board                             object
BBL                                        float64
Borough                                     object
X Coordinate (State Plane)                 float64
Y Coordinate (State Plane)                 float64
Open Data Channel Type                      object
Park Facility Name                          object
Park Borough                                object
Vehicle Type                                object
```

```
Taxi Company Borough                      object
Taxi Pick Up Location                     object
Bridge Highway Name                       object
Bridge Highway Direction                  object
Road Ramp                                 object
Bridge Highway Segment                    object
Latitude                                 float64
Longitude                                float64
Location                                  object
dtype: object
```

[5]:

```
df_2020.head()
```

[5]:

```
   Unique Key Created Date         Closed Date Agency  \
0    45289558   2020-01-01 2020-01-15 00:00:01  DOHMH
1    45288728   2020-01-01 2020-01-02 00:00:01  DOHMH
2    45288240   2020-01-01 2020-01-02 00:00:01  DOHMH
3    45287907   2020-01-01 2020-01-02 00:00:01  DOHMH
4    45285651   2020-01-01 2020-01-02 00:00:01  DOHMH

                               Agency Name  Complaint Type Descriptor  \
0  Department of Health and Mental Hygiene  Food Poisoning  3 or More
1  Department of Health and Mental Hygiene  Food Poisoning     1 or 2
2  Department of Health and Mental Hygiene  Food Poisoning     1 or 2
3  Department of Health and Mental Hygiene  Food Poisoning  3 or More
4  Department of Health and Mental Hygiene  Food Poisoning     1 or 2

                Location Type  Incident Zip      Incident Address  …  \
0       Other (Explain Below)       11215.0      625 UNION STREET  …
1  Restaurant/Bar/Deli/Bakery       11225.0   985 NOSTRAND AVENUE  …
2  Restaurant/Bar/Deli/Bakery       11385.0  1717 CORNELIA STREET  …
3  Restaurant/Bar/Deli/Bakery       11214.0    1602 SHORE PARKWAY  …
4  Restaurant/Bar/Deli/Bakery       10458.0   2701 DECATUR AVENUE  …

  Vehicle Type Taxi Company Borough Taxi Pick Up Location Bridge Highway Name  \
0          NaN                  NaN                   NaN                 NaN
1          NaN                  NaN                   NaN                 NaN
2          NaN                  NaN                   NaN                 NaN
3          NaN                  NaN                   NaN                 NaN
4          NaN                  NaN                   NaN                 NaN

  Bridge Highway Direction Road Ramp Bridge Highway Segment   Latitude  \
0                      NaN       NaN                    NaN  40.677963
1                      NaN       NaN                    NaN  40.664422
2                      NaN       NaN                    NaN  40.700366
3                      NaN       NaN                    NaN  40.595653
4                      NaN       NaN                    NaN  40.864866
```

```
   Longitude                                 Location
0 -73.984436  (40.677963041857886, -73.98443609121443)
1 -73.950982   (40.66442190467239, -73.95098201556382)
2 -73.905438  (40.700366489799876, -73.90543829006366)
3 -74.000173   (40.59565343138651, -74.00017283917487)
4 -73.888783   (40.86486556770799, -73.88878325729915)

[5 rows x 41 columns]
```

1.1.4 Step 3: Date counts

Create a DataFrame called `date_counts` that has the count of complaints per Complaint Type per day, then display it.

[6]:

```
date_counts = df_2020.groupby(['Complaint Type']).resample('D',on='Created␣
 ↪Date').size().reset_index(name='count_requests')
date_counts
```

[6]:

```
                Complaint Type Created Date  count_requests
0                    APPLIANCE   2020-01-01               6
1                    APPLIANCE   2020-01-02              27
2                    APPLIANCE   2020-01-03              29
3                    APPLIANCE   2020-01-04              11
4                    APPLIANCE   2020-01-05              12
…                          …          …             …
12750             Window Guard   2020-03-22               0
12751             Window Guard   2020-03-23               0
12752             Window Guard   2020-03-24               0
12753             Window Guard   2020-03-25               1
12754   X-Ray Machine/Equipment   2020-01-15               1

[12755 rows x 3 columns]
```

1.1.5 Step 4: Plotting over time

Create a line chart of the count of complaints over time, one line per `Complaint Type`.

[7]:

```
# your code here
fig=px.line(date_counts,x='Created Date', y='count_requests',color='Complaint␣
 ↪Type')
fig.show()
```

This has the information we need, but is a lot to look at. Let's only show complaint types that changed greatly (in March 2020) relative to the same period in the previous year (March 2019).

1.1.6 Step 5: March 2020 counts

Create a DataFrame called `mar_counts` that has the count of each `Complaint Type` in March 2020 in a column called `2020`. Use `.to_frame()` (instead of `.reset_index()`) to use the `Complaint Type` as the index. It should end up looking something like this:

Complaint Type	2020
APPLIANCE	824
Abandoned Vehicle	2500
Air Quality	657
…	…

Note there is no numeric index.

[8]:
```
# your code here
date_counts['month']=date_counts['Created Date'].dt.month
date_counts
mar=date_counts[date_counts['month']==3]
mar
```

[8]:
```
      Complaint Type Created Date  count_requests  month
60         APPLIANCE   2020-03-01               5      3
61         APPLIANCE   2020-03-02              21      3
62         APPLIANCE   2020-03-03              15      3
63         APPLIANCE   2020-03-04              15      3
64         APPLIANCE   2020-03-05              35      3
…                …          …                 …    …
12749   Window Guard   2020-03-21               0      3
12750   Window Guard   2020-03-22               0      3
12751   Window Guard   2020-03-23               0      3
12752   Window Guard   2020-03-24               0      3
12753   Window Guard   2020-03-25               1      3

[4109 rows x 4 columns]
```

[9]:
```
mar_counts=mar.groupby(['Complaint Type'])['count_requests'].sum().
 ↪to_frame(name='2020')
mar_counts
```

[9]:
```
                              2020
Complaint Type
APPLIANCE                      806
Abandoned Vehicle             2468
Air Quality                    643
Animal Facility - No Permit      3
Animal in a Park               195
…                             …
```

```
WATER LEAK                    1219
Water Conservation             151
Water Quality                   88
Water System                  2963
Window Guard                     2

[150 rows x 1 columns]
```

1.1.7 Step 6: Get March 2019 data

Follow Steps 0-2 again, this time with 311 requests for all of March 2019. Name the DataFrame `mar_2019`.

Similar to Step 0, if having trouble downloading, you can load from:

https://storage.googleapis.com/python-public-policy/data/311_mar_2019.csv.zip

[10]:

```python
# your code here
mar_2019= pd.read_csv(
    "https://storage.googleapis.com/python-public-policy/data/311_mar_2019.csv.
 ↪zip"
)
```

```
/tmp/ipykernel_109/1436439550.py:2: DtypeWarning:

Columns (8,31) have mixed types. Specify dtype option on import or set
low_memory=False.
```

1.1.8 Step 7: March 2019 counts

1. Get the `Complaint Type` counts for March 2019.
2. Add these to the `mar_counts` DataFrame as a column called 2019.
   - Reminder that adding a Series as a new column to a DataFrame matches rows based on the index.

[11]:

```python
import pandas as pd
import plotly.express as px
```

[12]:

```python
# your code here
mar_19=mar_2019.groupby(['Complaint Type']).size().reset_index(name='2019')
mar_19
```

[12]:

```
                    Complaint Type  2019
0                        APPLIANCE  1042
1                 Abandoned Vehicle     1
2                   Advocate - Other     8
3             Advocate-Business Tax     2
4     Advocate-Co-opCondo Abatement     2
```

```
..                          ...   ...
201             Water Conservation   298
202                  Water Quality   108
203                   Water System  3880
204                   Window Guard     1
205        X-Ray Machine/Equipment     1

[206 rows x 2 columns]
```

[13]:
```
mar_counts=pd.merge(left=mar_counts, right=mar_19,left_on='Complaint Type',␣
 ↪right_on='Complaint Type')
mar_counts
```

[13]:
```
                 Complaint Type  2020  2019
0                     APPLIANCE   806  1042
1             Abandoned Vehicle  2468     1
2                   Air Quality   643   642
3    Animal Facility - No Permit     3    10
4               Animal in a Park   195   211
..                          ...   ...   ...
133                  WATER LEAK  1219  2603
134          Water Conservation   151   298
135               Water Quality    88   108
136                Water System  2963  3880
137                Window Guard     2     1

[138 rows x 3 columns]
```

1.1.9 Step 8: Percent change

Use `mar_counts` to calculate the percent change from March 2019 to March 2020 for each `Complaint Type`. Save as the `pct_change` column. Should result in something like this:

Complaint Type	2020	2019	pct_change
APPLIANCE	824	1042	-0.20
Abandoned Vehicle	2500	1	2499.00
Air Quality	657	642	0.02
...	...	...	...

[14]:
```
# your code here
mar_counts['pct_change']=(mar_counts['2020']-mar_counts['2019'])/
 ↪mar_counts['2019']
mar_counts['pct_change']=mar_counts['pct_change'].apply(lambda x: float('{:.
 ↪2f}'.format(x)))
mar_counts
```

[14]:

```
                       Complaint Type  2020  2019  pct_change
0                           APPLIANCE   806  1042       -0.23
1                    Abandoned Vehicle  2468     1     2467.00
2                          Air Quality   643   642        0.00
3         Animal Facility - No Permit     3    10       -0.70
4                     Animal in a Park   195   211       -0.08
..                                ...   ...   ...         ...
133                        WATER LEAK  1219  2603       -0.53
134                Water Conservation   151   298       -0.49
135                     Water Quality    88   108       -0.19
136                      Water System  2963  3880       -0.24
137                      Window Guard     2     1        1.00

[138 rows x 4 columns]
```

1.1.10 Step 9: Filter

Filter to `Complaint Types` that both:

- Occurred at least 50 times in March 2020
- Changed (increased *or* decreased) by more than 90%

and save the DataFrame as `top_changed`. A couple of things that may be helpful:

- Selecting Subsets of Data in Pandas, starting from “Multiple condition expression”
- Getting absolute values

[17]:

```
# your code here
top_changed = mar_counts[(mar_counts['2020']>=50)&(mar_counts['pct_change'].
 ↪abs()>0.90)]
top_changed
```

[17]:

```
                       Complaint Type  2020  2019  pct_change
1                    Abandoned Vehicle  2468     1     2467.00
20                  Consumer Complaint  9324  1228        6.59
34                       Drug Activity   267   134        0.99
58          Homeless Person Assistance  1475   728        1.03
78                  Noise - Helicopter   368    96        2.83
84         Non-Emergency Police Matter  1846   751        1.46
104               Sanitation Condition   157  3040       -0.95
129                 Urinating in Public    63    33        0.91
```

1.1.11 Step 10: Top changed

Filter the `date_counts` to only the `top_changed` `Complaint Types`. Save as `top_changed_by_day`.

[18]:

```
# your code here
top_changed_by_day = date_counts[date_counts['Complaint Type'].
 ↪isin(top_changed['Complaint Type'])]
```

```
top_changed_by_day
```

[18]:

```
             Complaint Type Created Date  count_requests  month
91         Abandoned Vehicle   2020-01-01              63      1
92         Abandoned Vehicle   2020-01-02             156      1
93         Abandoned Vehicle   2020-01-03             151      1
94         Abandoned Vehicle   2020-01-04              82      1
95         Abandoned Vehicle   2020-01-05              97      1
…                        …            …               …     …
12068   Urinating in Public   2020-03-26               2      3
12069   Urinating in Public   2020-03-27               0      3
12070   Urinating in Public   2020-03-28               2      3
12071   Urinating in Public   2020-03-29               4      3
12072   Urinating in Public   2020-03-30               5      3

[722 rows x 4 columns]
```

1.1.12 Step 11: Plotting changed complaints

Make a similar plot to Step 4, but with only the top complaints (`top_changed_by_day`).

[23]:

```
# your code here
fig=px.line(top_changed_by_day, x="Created Date", y="count_requests",␣
 ↪color='Complaint Type')
fig.show()
```

1.2 Question 0

Did the change of any of the `Complaint Types` ***in Step 10/11 surprise you? Why or why not? (Speak at least one specifically.)*** The comsumer complaint from the beginning of March 2020 increased sharply (violations for price gouging during the coronavirus outbreak), then decreased to nearly the previous level at the end of March 2020(Department of Consumer and Worker Protection Issues Emergency Rule That Makes Price Gouging Illegal for Any Item or Service Needed to Limit the Spread of Coronavirus)

Then, give these a read:

- NY Daily News article
- Press release from Department of Consumer and Worker Protection

Overall caveat for this assignment: **correlation does not imply causation**.

1.3 Bonus: Charting against COVID-19 case counts

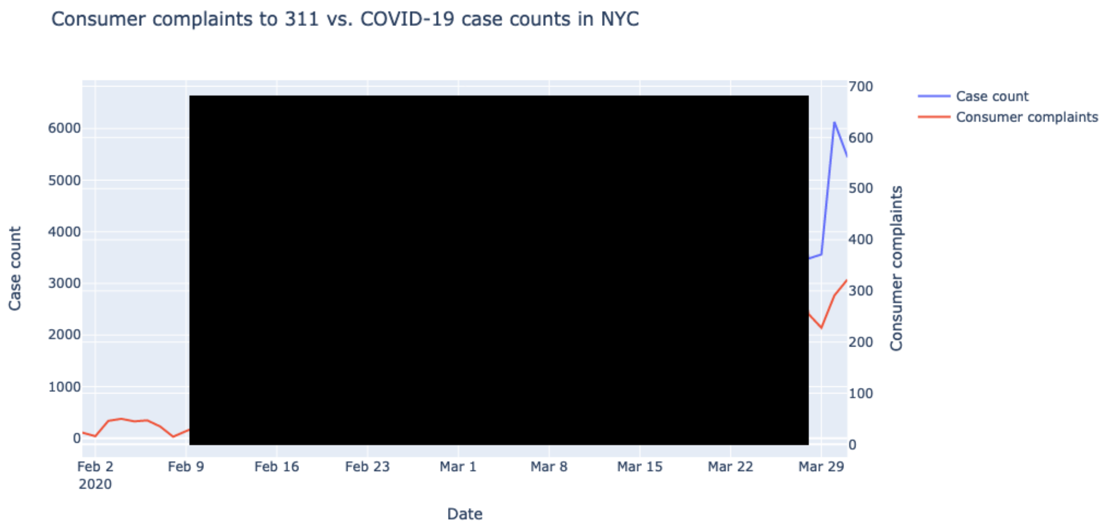
0.4 points

Let's take a look at the `Consumer Complaints` against the COVID-19 case numbers in NYC in the same graph. You'll need to:

1. Find data that provides the COVID-19 case counts for NYC by day.

2. Create a DataFrame with only the `Consumer Complaint Complaint Type` counts, by day.
3. Chart the two against each other for February through March.

The result should look something like this (without the black box):



Some resources that may be helpful:

- Reading CSV data from GitHub
- Two Y Axes in plotly
  - Note that the `plotly.graph_objects` syntax is a bit different than the `plotly.express` syntax we've been using. With `go.Scatter()`, you don't provide the DataFrame and the names of the columns; you pass `x` and `y` as lists/Series of the values themselves.
- Setting the Range of Axes Manually in plotly

[ ]:
```
# your code here
```

1.4 Bonus Question 1: What observations do you have?

YOUR RESPONSE HERE

Now turn in the assignment.

1.5 Tutorials

In the videos below, don't get hung up on mentions of JavaScript, Node.js, or Twilio — those were technologies used for another course.

1. Watch:
   1. What are APIs?
   2. APIs, Conceptually
2. Read Understanding And Using REST APIs
3. Watch:
   1. Let's look at some data

    2. Data formats
    3. API documentation
4. Read Python's Requests Library (Guide) through `The Message Body`

1.6 Participation

Reminder about the between-class participation requirement.