New York Crime Dashboard

Team Panini Head

Data Science Capstone Project
Data Acquisition and Pre-Processing Report

Date:

04/27/2021

Team Members:

Name: Ambrose Karella

Name: Janam Patel

Name: Naimish Bizzu

Identifying Data

Data Sources:

We got our data from NYC Open Data's website. The data is provided by the Police Department (NYPD) so, the data is correct and authentic. The website allowed different ways of extracting data from downloading in the CSV or GEOSON format to using the API calls to the data in the form of CSV or JSON. We chose to gather data in JSON format with the help of their great API. We can download the file but, we thought that getting data using API would be easier since we wouldn't have to download a big CSV file and then try to push that file onto our database.

Acquisition Process:

We know that the dataset is updated annually so whatever data is available on the NYC OpenData website is the final data until they upload or make changes to data at the end of the year.

As we said earlier, we had a choice of downloading data, but we chose the method of API to get our data. The data is public and doesn't require a private API key and there is no hard limit on rate as well. We wrote a simple code in python which allowed us to pull data. We originally wrote a simple for loop which would get data by offsetting each time, but it was taking a lot of time to get all the data. So, to solve this we used concurrent futures modules which would run code on different cores of the machine, and this sped up the process by a lot. Instead of spending close to 4 hours, we were able to get all 5 million plus rows in just the amount of 20 minutes, and this could only get better, if you have a heavy-duty computer with a higher number of cores.

As of right now, the code will only run on an IDE because we used concurrent futures modules which works with your local machine's multiple cores, and Jupyter Notebook didn't work well with concurrent futures at the time.

Thankfully, all the data is available in the one source, so we didn't have to worry about getting data from multiple sources and merging them after.

Issues:

We had three hurdles time, limit on rate of each pull and data format issues with server.

When pushing the data to server it'd mess with data and overwrite in other cells. We were able to fix this by removing problematic attribute that wasn't necessary.

With time, the problem was that there was 5,001,200 records, and if we had offset by 1000, then the simple for loop would need to run for 5012 times and after each iteration, we would have to merge to the previously collected data which was time consuming and also space consuming. To solve this, we used concurrent futures modules which did the job perfectly, we still used loop but it was much faster. The problem with limit on rate for each pull was that the API would only allow us to pull 1000 items only per call. So, the first API call would give us 0-999 items, but to get the 1000-1999, we would need to add offset option to our API link which would get us the next 1000 records of data. This was also easily fixed in the loop where we would just multiply loop counter with 1000 and offset it by that number.

Data-Processing

Even though the data is in good condition to use, we believe that pre-processing is required. With the number of nulls we have, we should establish which values we absolutely need based on the needs of our project.

Since we would like to map crime and look at crime historically, "latitude", "longitude", and "arrest_date" are non-negotiable. Thankfully, at most, we should only lose 28 rows at most.

We also notice that we have “x_coord_cd” and “y_coord_cd” but we also have latitude (‘lat’) and longitude (‘long’) which could mean that they’re repeated but after some research, we found that the “x_coord_cd” and “y_coord_cd” are just state plane coordinate system, which is an outdated way to lookup locations, so those columns can be dropped as well.

There is a column that supposedly has zip codes, but it doesn’t look right since the column contains values like “1” and “26001” which aren’t valid zip codes, so that column will be dropped as well.

There are nine null values for arrest_boro and jurisdiction_code which can be dropped as well.

With age_group column, there are only 16 missing values so we can drop those as well. But it also had a lot of values that don’t make sense like “320” or “1942” or “1937”, so unless someone whos lived for more than 100 years, we will drop the age_group values that don’t seem correct. Thankfully we’d only be dropping only 136 rows.

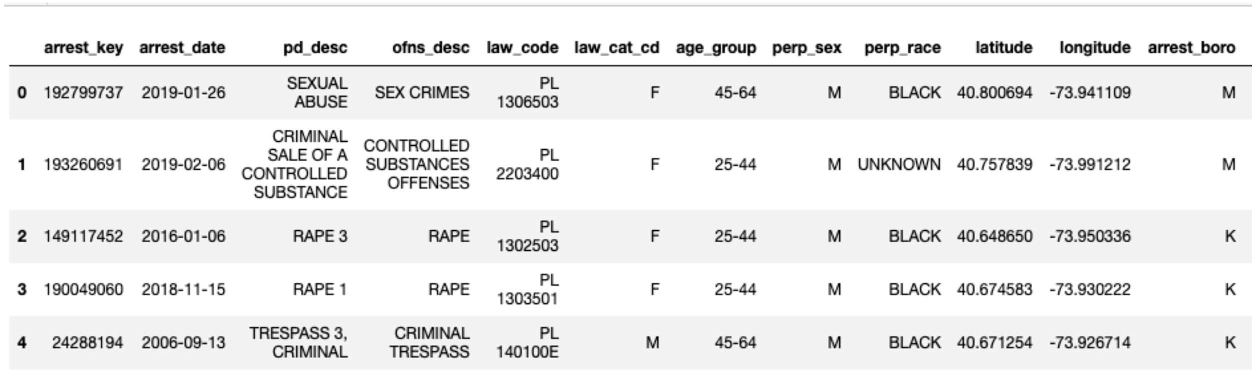
Another category that needed attention was the description of crimes. Close to 9000 crimes were listed without a description. Thankfully, using the law codes and cross-referencing with the NY penal code, we were able to recover all but 127 rows. These rows contained laws outside the NY penal code and would be a lot more difficult to look up.

In all, we were able to preserve most of the rows. Almost all the lost data consisted only of duplicates.

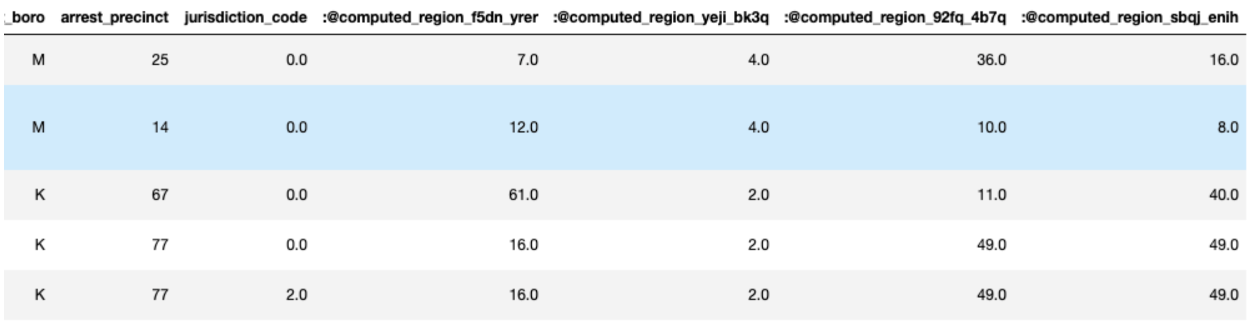
Appendix

The code used for data gathering is written in the **crimeCollector_V2.py** file and preprocessing code is written in the **Preprocessing.ipynb** file and both are attached to view the code.

Here’s a peak of the data



	arrest_key	arrest_date	pd_desc	ofns_desc	law_code	law_cat_cd	age_group	perp_sex	perp_race	latitude	longitude	arrest_boro
0	192799737	2019-01-26	SEXUAL ABUSE	SEX CRIMES	PL 1306503	F	45-64	M	BLACK	40.800694	-73.941109	M
1	193260691	2019-02-06	CRIMINAL SALE OF A CONTROLLED SUBSTANCE	CONTROLLED SUBSTANCES OFFENSES	PL 2203400	F	25-44	M	UNKNOWN	40.757839	-73.991212	M
2	149117452	2016-01-06	RAPE 3	RAPE	PL 1302503	F	25-44	M	BLACK	40.648650	-73.950336	K
3	190049060	2018-11-15	RAPE 1	RAPE	PL 1303501	F	25-44	M	BLACK	40.674583	-73.930222	K
4	24288194	2006-09-13	TRESPASS 3, CRIMINAL	CRIMINAL TRESPASS	PL 140100E	M	45-64	M	BLACK	40.671254	-73.926714	K



_boro	arrest_precinct	jurisdiction_code	:@computed_region_f5dn_yrer	:@computed_region_yeji_bk3q	:@computed_region_92fq_4b7q	:@computed_region_sbqj_enih
M	25	0.0	7.0	4.0	36.0	16.0
M	14	0.0	12.0	4.0	10.0	8.0
K	67	0.0	61.0	2.0	11.0	40.0
K	77	0.0	16.0	2.0	49.0	49.0
K	77	2.0	16.0	2.0	49.0	49.0

Table of Contributions

The table below identifies contributors to various sections of this document.

	Section	Writing	Editing
1	**Data Sources**	**Janam**	**Ambrose**
2	**Data Pre-Processing**	**Ambrose**	**Ambrose, Janam**
3	**Appendix**	**Naimish**	**Naimish**

Grading

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.