NYU

Big Data Final Presentation Group 23

Github Repository:
https://github.com/Zankar100/BigDataGroup23

PRESENTED BY: Bansi Shah (bks7385), Zankar Murudkar (zm2117), Tirth Patel (tmp9936)
DATE: 12/13/2021

Agenda

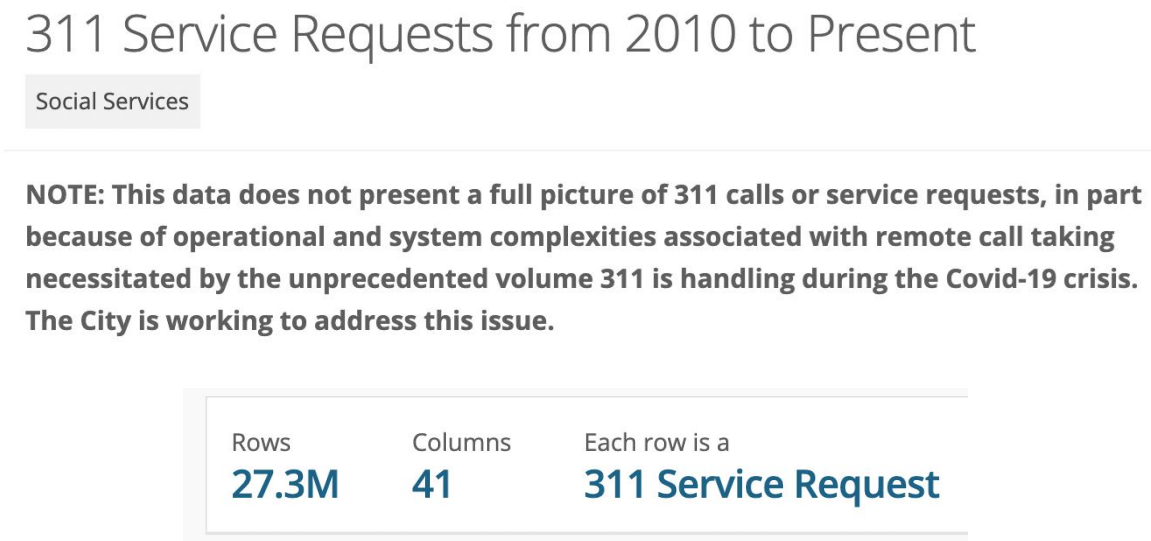
- Tools used for Profiling and Cleaning
- Dataset
- Data Issues and Methods to Clean them
- Challenges

Tools Used for Profiling and Cleaning

VIDA-NYU/
openclean

openclean - Data Cleaning and data profiling library for Python

Dataset



https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

Data Issues and Methods to Clean them

Null Values Count

Unique_Key=0.0,
Created_Date=0.0,
Closed_Date=0.04949,
Agency=0.0,
Agency_Name=0.0,
Complaint_Type=0.0,
Descriptor=0.0032593333333333333,
Location_Type=0.305324,
Incident_Zip=0.3192426666666667,
Incident_Address=0.34485266666666664,
Street_Name=0.34491133333333335,
Cross_Street_1=0.5335506666666666,
Cross_Street_2=0.54303,
Intersection_Street_1=0.6206986666666666,
Intersection_Street_2=0.6230033333333334,
Address_Type=0.52,
City=0.3392153333333333,
Landmark=0.762316,
Facility_Type=0.44323,
Status=0.0, Due_Date=0.682708,
Resolution_Description=0.024336,
Resolution_Action_Updated_Date=0.05073866666666667
Community_Board=0.012646,
BBL=0.5459406666666666,
Borough=0.012646,
X Coordinate (State Plane)=0.47851666666666665,
Y Coordinate (State Plane)=0.47848066666666667,
Open Data Channel Type=1.2666666666666667e-05,
Park Facility Name=1.2666666666666667e-05,
Park_Borough=0.012646,
Vehicle Type=0.9994933333333333,
Taxi Company Borough=0.9989966666666666,
Taxi Pick Up Location=0.9869713333333333,
Bridge Highway Name=0.9947166666666667,
Bridge Highway Direction=0.9948233333333333,
Road Ramp=0.9948666666666667,
Bridge Highway Segment=0.9946873333333334,
Latitude=0.47852466666666665,
Longitude=0.47852466666666665,
Location=0.47851266666666664)

Column Dependency (Dates)

- We have 2 columns ‘Created Date’ and ‘Closed Date’.
- For the data to be valid, ‘Closed Date’ must be greater than ‘Created Date’
- We first convert these columns from string type to timestamp and then compare them to deem valid or invalid ‘Closed Date’.
- The rows with ‘Closed Date’ earlier than ‘Open Date’ are invalid and hence removed from the data.

```
('Created_Date',to_timestamp(df_data.Created_Date,"MM/dd/yyyy hh:mm:ss a"))
('Closed_Date',to_timestamp(df_data.Closed_Date,"MM/dd/yyyy hh:mm:ss a"))
('Due_Date',to_timestamp(df_data.Closed_Date,"MM/dd/yyyy hh:mm:ss a"))
```

```
[ ] df_new = df_data.filter(df_data.Closed_Date > df_data.Created_Date)

df_new.count()

25117976

[ ] df_new2 = df_data.filter(df_data.Closed_Date.isNull())

[ ] df_new2.count()

756719
```

Abbreviate Agency Name

Takes in an agency name and converts to abbreviated form, if the agency is the Office of Special Enforcement

```
dstemp.Agency.unique()
```

```
array(['NYPD', 'DOT', 'DPR', 'DOHMH', 'DOB', 'DEP', 'DSNY', 'DOF', 'DHS',
       'HPD', 'EDC', 'DCA', 'TLC', 'DOITT', 'DFTA', 'DOE', 'TAX', 'DCAS',
       'TAT', 'MAYORâ\x80\x99S OFFICE OF SPECIAL ENFORCEMENT', 'ACS',
       'DVS', 'NYCEM', 'HRA', 'FDNY', 'DCP', 'DORIS', '3-1-1'],
      dtype=object)
```

```
array(['NYPD', 'DOT', 'DPR', 'DOHMH', 'DOB', 'DEP', 'DSNY', 'DOF', 'DHS',
       'HPD', 'EDC', 'DCA', 'TLC', 'DOITT', 'DFTA', 'DOE', 'TAX', 'DCAS',
       'TAT', 'MOSE', 'ACS', 'DVS', 'NYCEM', 'HRA', 'FDNY', 'DCP',
       'DORIS', '3-1-1'], dtype=object)
```

Street Name Misspelled

After clustering we found that some street names were misspelled.

We can replace the street names to solve this issue

```
Cluster 2
  ST JOHNS AVENUE (x 143)
  ST JOHN'S AVENUE (x 10)
  AVENUE ST JOHNS (x 38)
  AVENUE ST JOHN'S (x 1)

Cluster 3
  CENTRAL PARK W (x 19)
  W CENTRAL PARK (x 4)
  W CENTRAL PARK PARK (x 4)
  CENTRAL PARK PARK W (x 1)

Cluster 4
  CENTRAL PARK WEST PARK W (x 1)
  CENTRAL WEST PARK W (x 4)
  CENTRAL PARK WEST. PARK W (x 1)
  W CENTRAL PARK WEST PARK W (x 5)
```

Agency Name Misspelled

```
Cluster 1
  SCHOOL - ARCHIMEDES ACADEMY FOR MATH, SCIENCE AND TECHNOLOGY APPLICATIONS (x 1)
  SCHOOL - ARCHIMEDES ACADEMY FOR MATH, SCIENCE, AND TECHNOLOGY APPLICATIONS (x 3)

Cluster 2
  SCHOOL - SCHOLARS ACADEMY (x 2)
  SCHOOL - SCHOLARS' ACADEMY (x 1)
```

Zip Code - Out of Range

```
len(pd_data.Incident_Zip.unique())
```

```
1378
```

Total unique zip codes in our data

We used reference data to check which zip codes are invalid and marked them as ‘unspecified’.

```
[ ]  DBSCANOutliers(eps=0.01).find(incident_zip)
```

```
'UNKOWN',
'98036-1542',
'55555',
'999999',
'031546',
"DON'T KNOW",
'4009-0517',
'1735-9100',
'L7R1H2',
'NJ 07310',
'14450 0457',
'N/A',
'.',
'46',
'000000000',
'NOT SURE',
'11226',
'UNKNOWN',
'115I0',
'NTY',
'920469046',
'NA',
'10467',
'10459',
'171052461',
'0000 0000',
'11111',
'ANONYMOUS',
'NO CLUE',
```

```
def check_zip(z):

    if z in zip_data.ZipCode.unique():
        return int(z)

    else:
        return 'Unspecified'
```

```
[11] pd_data.Incident_Zip = pd_data.Incident_Zip.map(lambda x: check_zip(x))
```

```
pd_data.Incident_Zip.unique()
```

```
array(['11421', '10468', '11214', '11229', '11420', '10469', '10456',
       '11217', 'Unspecified', '11357', '11369', '11215', '11210',
       '11233', '10034', '11103', '11220', '11368', '11361', '10016',
       '10463', '11366', '10032', '11419', '10457', '11207', '11226',
       '11219', '11379', '10025', '11435', '11212', '11236', '11230',
       '10462', '11105', '10031', '11208', '10002', '10019', '11209',
       '10459', '10312', '11205', '10309', '10451', '10021', '11203',
       '11102', '10465', '11355', '10460', '10314', '10464', '11218',
       '11414', '11385', '11204', '11206', '11237', '10029', '10301',
       '10452', '10023', '10022', '11691', '10466', '10473', '11373',
       '10075', '10003', '10011', '11358', '10454', '11378', '10305',
       '11377', '11375', '10012', '11211', '11429', '10040', '11432',
       '11106', '11231', '11238', '11234', '11372', '11356', '10467',
       '11235', '11223', '11213', '11370', '10001', '10065', '10014',
       '10461', '11413', '11411', '11694', '11365', '11418', '11216',
       '10475', '11374', '10013', '10026', '11222', '11201', '10471',
       '11104', '11225', '11423', '10472', '10028', '10036', '11232',
       '10004', '11427', '10038', '10458', '10027', '10009', '10017',
       '11434', '10455', '10307', '11428', '10306', '11436', '10037',
       '10453', '10005', '10018', '11221', '11224', '11228', '10303',
       '10128', '10304', '11433', '10024', '11692', '10035', '11101',
       '10302', '11417', '11004', '11354', '11426', '11693', '11412',
       '10007', '10308', '11415', '10030', '11422', '10474', '10470',
       '10033', '10010', '11360', '11416', '11363', '11364', '10039',
       '10310', '11367', '11362', '10280', '10020', '11239', '10006',
       '11697', '10044', '11359', '11005', '11695'], dtype=object)
```

Mapping Zip Code to Borough

- After validating all the zip codes, we check if the specified zip code value actually belongs to the specified borough or not.
- We used reference data for this and compared its (borough, zip code) pair to the (borough, zip code) pair in our data.
- If the data did not match then we replaced the borough that matches the given zip code.

```
[ ]  for i in range(len(specified_zip_data)):
       if (specified_zip_data.Borough[i] == zip_data.loc[zip_data['ZipCode'] == specified_zip_data.Incident_Zip[i], 'Borough'].iloc[0]):
         continue
       else:
         specified_zip_data.Borough[i] = zip_data.loc[zip_data['ZipCode'] == specified_zip_data.Incident_Zip[i], 'Borough'].iloc[0]
```

```
boroughs = ds.distinct('Borough')
for rank, val in enumerate(boroughs.most_common()):
    st, freq = val
    print(f'{rank + 1:<3} {st}  {freq:>10,}')
```

```
1   BROOKLYN   1,381,026
2   QUEENS   1,278,847
3   MANHATTAN   1,114,528
4   BRONX     759,217
5   STATEN ISLAND     262,599
6   Unspecified     184,798
7         18,985
```

```
[ ]  specified_zip_data.Borough.unique()
```

```
array(['MANHATTAN', 'BROOKLYN', 'BRONX', 'STATEN ISLAND', 'QUEENS'],
      dtype=object)
```

Community Board - Out of Range

```
len(community_data.cd_short_title.unique())
```

```
59
```

Total no. of valid Community Boards

```
len(pd_data.Community_Board.unique())
```

```
80
```

Total no. of unique Community Boards found in our data.

We used reference data to check which values in our data column are invalid and termed them as ‘unspecified’.

Cont.

Difference in Reference data and dataset

```
community_data.cd_short_title.head(10)
```

```
0     Manhattan CD 1
1     Manhattan CD 2
2     Manhattan CD 3
3     Manhattan CD 4
4     Manhattan CD 5
5     Manhattan CD 6
6     Manhattan CD 7
7     Manhattan CD 8
8     Manhattan CD 9
9    Manhattan CD 10
Name: cd_short_title, dtype: object
```

```
pd_data.Community_Board.head(10)
```

```
0         09 QUEENS
1          08 BRONX
2       11 BROOKLYN
3       15 BROOKLYN
4         10 QUEENS
5         10 QUEENS
6          11 BRONX
7          04 BRONX
8       06 BROOKLYN
9    0 Unspecified
Name: Community_Board, dtype: object
```

Range check on Community Board

```
def check_community(z):

  if z in community_data.cd_short_title.unique():
        return z

  else:
        return 'Unspecified'
```

```
pd_data.Community_Board = pd_data.Community_Board.map(lambda x: check_community(x))
```

Total no. of unique community board values in the data after cleaning.

```
len(pd_data.Community_Board.unique())
```

```
60
```

Data cleaning on Datasets with Overlapping Columns

DYCD after-school programs: Immigrant Services	https://data.cityofnewyork.us/Social-Services/DYCD-after-school-programs-Immigrant-Services/zmut-au2w
Board of Standards and Appeals	https://data.cityofnewyork.us/w/yvxd-uipr/25te-f2tw?cur=AbBzeB8m2eb&from=root
3K Projects by Site Locations	https://data.cityofnewyork.us/Education/3K-Projects-by-Site-Locations/yv4m-nu6d
Fire Incident Dispatch Data	https://data.cityofnewyork.us/Public-Safety/Fire-Incident-Dispatch-Data/8m42-w767
Jobs Created by Energy Cost Savings Program Savings for Businesses - FY2020	https://data.cityofnewyork.us/City-Government/Jobs-Created-by-Energy-Cost-Savings-Program-Saving/yqky-aebb

Thank you!