Open Data Profiling, Quality and Analysis

Chinmay Wyawahare
New York University Tandon School of Engineering
Brooklyn, New York
chinmay.wyawahare@nyu.edu

HemanthTeja Yanambakkam
New York University Tandon School of Engineering
Brooklyn, New York
hy1713@nyu.edu

Vineet Viswakumar
New York University Tandon School of Engineering
Brooklyn, New York
vv913@nyu.edu

ABSTRACT

Often, data scientists, machine learning engineers and big data specialists find it hard to identify sources of high quality data. Data available in public domains are often riddled with data quality issues such as missing values, missing metadata, non standard entries for column values, etc. In this project, we go through the data available at the NYC Open Data site and anticipate and fix the issues mentioned above in these datasets.

KEYWORDS

Data profiling, data mining, big data

ACM Reference Format:
Chinmay Wyawahare, HemanthTeja Yanambakkam, and Vineet Viswakumar. 2019. Open Data Profiling, Quality and Analysis . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data available in public domains are often riddled with data quality issues such as missing values, missing metadata, non standard entries for column values, etc. Whenever we start off with a public dataset, many-a-times it's the case when the data hasn't undergone data cleaning or data preprocessing. As a result, we have dataset containing NaNs of zeros or sometimes erroneous headers as well. In order to use the dataset and perform analysis over such dataset, we follow standard methods of data cleaning, data pre-processing and data profiling to make the data usable for production level usage.

The dataset which we have used for this project consisted of 1900 csv files comprising over 37 GBs of data. In order to process this vast amount of data, we used New York University's High Performance Computing 48-node DUMBO cluster and AWS t2.2xlarge instance to handle the heavy computations. Using bigdata technologies like Spark, Pandas and Modin (Distributed pandas) to work on dataframes comprising the dataset, we calculated metadata for each dataset and each column to identify misclassified entries. With the metadata information, we computed statistics to look at the data distribution and report the data types like integer, float, string, date. Along with this, we calculated precision and recall as metric for our algorithm to identify the labels from NYC Columns datasets using Fuzzy logic and regular expressions (Regex).

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PROJECT TASKS

2.1 Task 1

In Task 1, we perform generic profiling as in real world open data rarely comes with well documented metadata. This makes it hard for developers and data scientists to identify the data type for column values and fix any issues there might be with the data. Lack of metadata also makes it hard to discern patterns that may be present in the dataset.

After importing the 1900 csv files, we processed each csv by creating a spark dataframe along with the datasets.csv file to find the mapping for each of the files. In order to find the various data types across each files, we defined user defined functions for each data type and computed the statistics per data types. Furthermore, we used *udf* functions on spark dataframe and *collected* the corresponding files to compute the statistics per each data type.

While performing this task, we encountered a major issue of typecasted data into the string column. We came across many entries which were of INTEGER, FLOAT and DATE data types but were typecasted into STRING datatype while data preparation. To make the analysis richer, we further took a deep drive into the STRING data type and computed the count for the datatypes - STRING/DATE, STRING/INTEGER, STRING/FLOAT to report the actual datatype count and breakdown the analysis for STRING datatype as well.

We were able to work with 1790 JSONs as we faced infrastructure issues with DUMBO cluster while processing the huge csv files with file sizes ranging from 1.8 gigabytes to 3.21 gigabytes. This was caused mainly due to java out of heap memory error as the default configured jvm heap size was insufficient to process the bigger files. In order to fix this infrastructure issue, we also used *AWS t2.2xlarge* instances with 8 cores and 32 gigabytes of RAM. It took about 7-10 days to dump all the JSONs from the csv files including for the larger datasets.

2.2 Task 2

In Task 2, we perform semantic profiling, i.e identify the data type for values in each column in the dataset and we count the number of occurrences for each semantic type. We have used NYCColumns

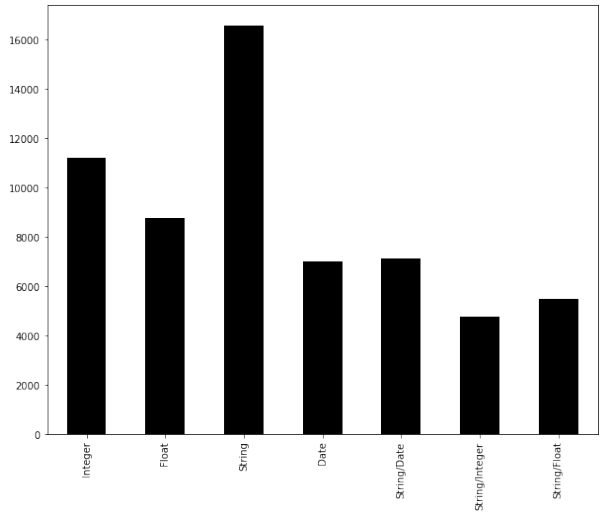

Figure 1: Distribution of data types across NYCOpenData

datasets with cluster2.txt files with the total file sizes comprising about 64 MB.

We used *distributed pandas dataframes* using *modin* framework which essentially uses *ray* and *dask* frameworks to use the distributed infrastructure and computing power. This task took about 3-4 hours to compute the semantic data types for each of the files from cluster2.txt. For infrastructure, we used *NYU HPC DUMBO cluster* and *AWS t2.2xlarge instance* to take advantage of the distributed cluster. We used libaries like fuzzywuzzy, modin to process the rows for pyspark dataframes.

3 METHODOLOGY

To identify the semantic types mentioned in section 1.2, we use a combination of techniques including string matching, regular expressions and dictionary match.

3.1 Fuzzy Logic

Fuzzy logic helps in approximate string matching. The idea of fuzzy logic is to find strings that approximately match a string pattern. The python library that helps us achieve this functionality is fuzzywuzzy. This library helps is computing the Levenshtein distance between a set of strings.

The Levenshtein distance is a metric to measure how similar two strings are by computing the distance between them. The idea of distance is slightly different though. This metric measures the least number or minimum number of edits that has to be performed on a word in order to change the word to a sequence that is similar to the word against which it is compared. This distance is computed in the form of a ratio. The Levenshtein distance ration between two strings – a and b , can be defined as follows:

$$\frac{(|a| + |b|) - lev_{a,b}(i,j)}{|a| + |b|}$$

Modulus function of the strings simply represents the length of the respective sequences. Furthermore, the fuzzywuzzy library can also help in smartly comparing and detecting strings with the help of "optimal partial" logic. This algorithm considers two strings of different lengths and seeks the best similarity matching score for a substring that is equal to the length of the first substring.

The fuzz token functions also help in providing an additional functionality of getting rid of lower case and punctuation. The "process" module of fuzzywuzzy also helps in calculating the highest string similarity when compared against a vector of strings. These functionalities helped us compute and analyse the semantic types and profile the data into respective types.

3.2 Regular Expressions

A regular expression is a combination of multiple characters that depict a special text for describing a search pattern. Every single meta-character in the regex has a special meaning that helps us yield a special search pattern. Regular expressions are processed by the regex processes that translate a regular expression into an internal representation. This representation is matched and executed against a string expression the text is being searched in. The regex module in python helps us achieve this functionality.

4 SEMANTIC PROFILING

4.1 Person Name

names come as either a combination of a First Name and Last Name or First Name, Middle Name and Last Name or the entire Full Name, eg. John Smith, Adam Elvis Presley, Vembu Srinivasan Raghav Hari Krishna.

Another issue when it comes to detecting names is that sometimes people have names of famous locations, deities, cardinal directions, colors, etc., eg.Kanye West.

A simple solution we implemented to solve this issue is run a regular expression and check if the value is a combination of alphabets and that the first letter is a capital letter as names don't contain numbers or special characters.

Business Name. Identifying names of businesses and companies is hard as they can easily be confused with names of people or places.

To identify if the column value is the name of a business, we run a regular expression to see if the column value has a registration value in its name, eg.LLC, Inc, lp., corp., co., co, ltd, capital, services, holdings.

4.2 Phone Number

In the US, phone numbers are a 10 digit combination. Each 3 digit area code has a capacity of 7,919,900 telephone numbers. Since phone numbers allow only numbers, we can easily identify if a given value is a phone number by checking if it is a 10 digit number.The disadvantage of using a regex to check if the value is a valid phone number is that in the US, the 5th and 6th digit cannot be 11 as these numbers are reserved for emergence services.

4.3 Address

An address represents the location of a building. As such, they include names of streets, avenues, zip codes, city and state names. To identify if the given value is an address, we compare the ratio of the occurrences of words like 'street','avenue', 'place' with the total number of words in that give value. If the ratio is above .85, we claim that the particular value is an address.One disadvantage of this method is that a full address can easily be confused for a street name.

4.4 Street Name

We use a similar technique compared to the technique we used in address identification to verify if the given value is a street name. Street name generally include words like 'street', 'avenue', 'road', 'boulevard', 'place' in their names.We compare the ratio of the occurrences of these words with the total number of words in the given entry. If the ratio is above 0.85, we claim that the value is a street name.

The advantages of this method is that we ignore the name and just search for words like 'street' or 'avenue'. This makes it less likely that the algorithm would confuse it for the name of a person as quite often, streets are named after famous personalities.

4.5 City

A city is defined as a large town with a local government. Cities, like places or locations are named after the people who discovered it or after some cultural significance and hence have no discernible pattern by which we can say that a given value is a city. To verify if a given value is a name of a city, we compare the value with a list of city names stored in a dictionary. If the name exists in the dictionary, we claim that the value is a city.

The advantage of this method is that the process is very quick. Searching in a dictionary is a fairly quick process and does not have any considerable memory overhead. The disadvantage of this method is that the dictionary might not contain the names of all the cities that exist. Hence, there is a probability that the value is not matched to an entry in the dictionary even though it is a city.

4.6 Neighborhood

A neighborhood is a geographically localised community. Neighborhoods can be combination of street names and addresses and can even include famous localities. To identify if a given value is a neighborhood, we create a list of known neighborhoods for the 5 boroughs in New York. We then check if the given value exists in the list we've built. If it exists, we claim that the value is the name of a neighborhood.

4.7 Coordinates

The geographical co-ordinate system can be used to pinpoint an exact location on a map. The range of values for coordinates vary from 90°N to 90°S for latitude and 180°E to 180°W for longitude. To identify if a given value is a coordinate, we run a regular expression to verify if the given value is a number and is lesser than 90 for latitude, or lesser than 180 for longitude.

4.8 Zip code

A ZIP code is a postal code used by the United States Postal Services. It is a 5-digit combination of numbers. The ZIP+4 code is a 9 digit combination of numbers which includes the 5 digit ZIP code and a 4 digit number representing a more specific location.To identify if a given value is a zip code, we run a regular expression to check if the given value contains 2 sets of numbers: a 5-digit number followed by a 4-digit number separated by a hyphen. If the value matches the regex pattern, we claim that the value is a ZIP code.

An advantage of using regex for ZIP codes is that it does not have the same restrictions as phone numbers in the US. This makes it quicker to identify if the given value is a ZIP code.

4.9 Borough

To verify if a given value is a borough, we create 2 lists. The 1st list contains values of counties and 2nd list contains the names of boroughs in New York City. For a value to be considered a borough, the value must be present in the counties list or the boroughs list. If the value is not present in either list, we claim that the value is not a borough.

4.10 School name

To verify if a given value is a school name, we compare the ratio of the similarity between the given value and a list of school names. If the ratio is above 0.85, we claim that the given value is a school name.

One disadvantage of this method is that the value might not be in the list even if it was the name of a school. This makes it impossible to detect which will leas to misclassification. Another potential drawback of this method is that it could be time consuming when implemented on large datasets.

4.11 Color

To verify if a given value is a color, we compare the ratio of the similarity between the given value and a list of colors. If the ratio is above 0.85, we claim that the given value is a color.

4.12 Car make

To verify if a given value is a car manufacturer, we compare the ratio of the similarity between the given value and a list of car manufacturers. If the ratio is above 0.85, we claim that the given value is the name of a car manufacturer.

4.13 City agency

City agencies help maintain different essential and non essential services to it's residents. To verify if a given value is a city agency, we compare the ratio of the similarity between the given value and a list of city agencies and their abbreviations. If the ratio is above 0.85, we claim that the given value is the name of a city agency.

4.14 Areas of study

To verify if a given value is an educational major, we compare the ratio of the similarity between the given value and a list of majors offered by universities and colleges. If the ratio is above 0.85, we claim that the given value is an area of study.
The advantage of this method is it's speed. Since we're searching for similar values in a dictionary(just similar, doesn't have to be exactly the same), the algorithm can match these patterns quickly.
The limitation of this strategy is that the list may not have all possible majors offered by universities. This may lead the algorithm to believe that a certain value is not a university major because it doesn't match any value in the list, while in reality, it is a major offered by universities.

4.15 Subjects in school

To verify if a given value is a school subject, we follow the same process as we did for academic majors, i.e we compare the ratio of the similarity between the given value and a list of subjects thought in schools. If the ratio is above 0.85, we claim that the given value is a school subject.
The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets.

4.16 School Levels

To verify if a given value is a school level, we compare the ratio of the similarity between the given value and a list of school levels. If the ratio is above 0.85, we claim that the given value is the name of a school level.
The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets.

4.17 College/University names

To verify if a given value is the name of a university, we compare the ratio of the similarity between the given value and a list of universities and colleges. If the ratio is above 0.85, we claim that the given value is the name of a University or College.
The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets. Another problem could arise as many universities or colleges can be named after famous personalities or places. By employing a dictionary mapping method, there could arise a situation where we claimed a value is a university when in reality it is not. e.g New York could be claimed to be a university as it bears resemblance with New York University while in reality, it the name of a state.

4.18 Websites

A website is an address where a user can find and host information on the internet. Websites generally start with 'www' and end with 'com', 'in', 'org', 'info', etc. To verify if a given value is a website, we run a regular expression to split the value by '.'. After splitting, if we find any of the values mentioned('www', 'http', 'https', 'com', 'org', 'gov', 'in', 'me', 'info', 'nyc', 'us') we claim that the value is a website.
An advantage of this method is that we save time by not comparing the entire string. Instead, we only search for patterns that must definitely exist in websites by using regular expressions and string matching.

4.19 Building Classification

To identify if the entry is a building classification, we first get a list of building codes and it's description from the NYC government website. We then compare the ratio of the similarity between the value and the list we've created. If the ration is greater than 0.85, we claim that it is a building classification.

4.20 Vehicle Type

Vehicle types can be broadly classified as vans, cars, buses, tractors, lorries, etc. To verify if a given value is a vehicle type, we compare the ratio of the similarity between the given value and a list of known vehicle types. If the ratio is above 0.85, we claim that the given value is the name of a vehicle type.
The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets.

4.21 Type of location

To verify if a given value is a location type, we compare the ratio of the similarity between the given value and a list of location types. If the ratio is above 0.85, we claim that the given value is the name of a type of location.
The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets.

4.22 Parks and Playgrounds

To verify if a given value is a park or a playground, we compare the ratio of the similarity between the given value and a list of known parks and playgrounds. If the ratio is above 0.85, we claim that the given value is the name of a park or playground.

The limitation of this strategy is that for every value, we need to check the entire list to find if there are similar entries. This can be a very inefficient method with regards to speed when it comes to large datasets.

5 PRECISION AND RECALL

5.1 Precision

Precision is measured over the total predictions of the model. It is the ratio between the correct predictions and the total predictions. In other words, precision indicates how good is the model at whatever it predicted.

Overall precision: 0.5945

5.2 Recall

Recall is the ratio of the correct predictions and the total number of correct items in the set. It is expressed as percentage of the total correct(positive) items correctly predicted by the model. In other words, recall indicates how good is the model at picking the correct items.

Overall recall: 0.7134

Sematic type	Precision	Recall
person_name	0.5868263473053892	0.7424242424242424
business_name	0.35294117647058826	0.5333333333333333
phone_number	0.6868686868686869	0.7555555555555555
address	0.6222222222222222	0.717948717948718
street_name	0.5757575757575758	0.6333333333333333
city	0.9342915811088296	0.9538784067085954
neighborhood	0.9	0.924
lat_lon_cord	0.9151670951156813	0.9442970822281167
zip_code	0.7528089887640449	0.8701298701298701
borough	0.7348484848484849	0.8899082568807339
school_name	0.5234375	0.6836734693877551
color	0.5816993464052288	0.7416666666666667
car_make	0.5823529411764706	0.8918918918918919
city_agency	0.6767676767676768	0.7528089887640449
area_of_study	0.17989417989417988	0.3434343434343434
subject_in_school	0.5113636363636364	0.5844155844155844
school_level	0.6043956043956044	0.7051282051282052
college_name	0.4	0.7924528301886793
website	0.25	1.0
building_classification	0.1348314606741573	0.21052631578947367
vehicle_type	0.8227848101265823	0.7222222222222222
location_type	0.8536585365853658	0.9545454545454546
park_playground	0.38636363636363635	0.6071428571428571
other	0.7115384615384616	0.8473282442748091

5.3 Task-2 Statistics

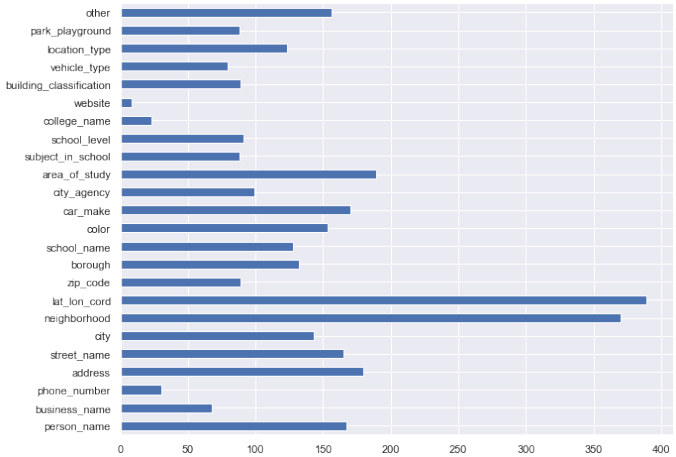

Figure 2: Distribution of labels across the NYCColumns data

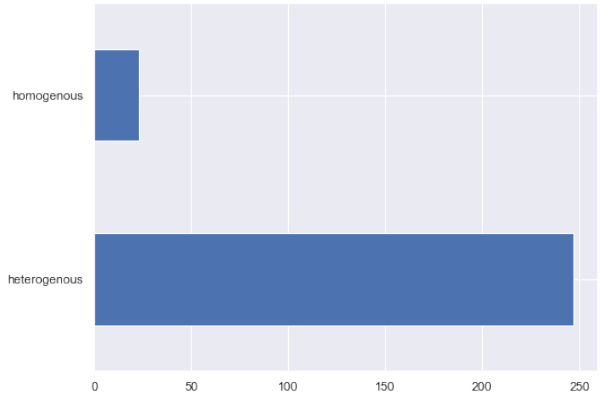

Figure 3: Distribution of nature of data

6 TASK 3

6.1 Hypothesis

For task 3, we chose to work on NYC 311 dataset and identify the three most frequent 311 complaint types by borough. We developed different visualizations like area charts, treemap, stacked bar graphs and so on to address questions like:

- Identify the three most frequent 311 complaint types by boroughs
- Are the same complaint types frequent in all five boroughs of the City?
- How might you explain the differences?
- How does the distribution of complaints change over time for certain neighborhoods and how could this be explained?

Q. Identify the three most frequent 311 complaint types by boroughs

We identify the most frequent complaint for each borough.

Queens:
In Queens, we find that the most frequent complaints are for residential noise, blocked driveways and street conditions.

Brooklyn:
In Brooklyn, the most frequent complaint is regarding residential noise.

Manhattan:
In Manhattan, the most frequent complaint is regarding residential noise.

Bronx:
In Bronx, the most frequent complaints are for residential noise and lack of heated water.

Staten Island:
In Staten Island, the most frequent complaint is regarding street conditions.

Q. How does the distribution of complaints change over time for certain neighborhoods and how could this be explained?

To understand the distribution of complaints and how they change over time, we build a stacked line graph where the X-axis represents time and the Y-axis represents the count of complaints. The colored area between two line graphs represents a particular complaint.

From the graph, we understand that the general trend of complaints is that the number of complaints have steadily increased from 2011-2018. However, there is a sharp decline in the number of complaints from 2019-2020. This can be due to lack of data for the year 2020.

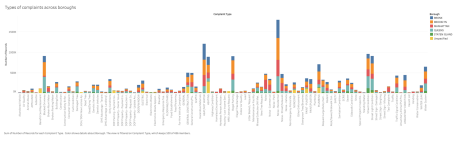

Figure 4: Types of complaints across boroughs

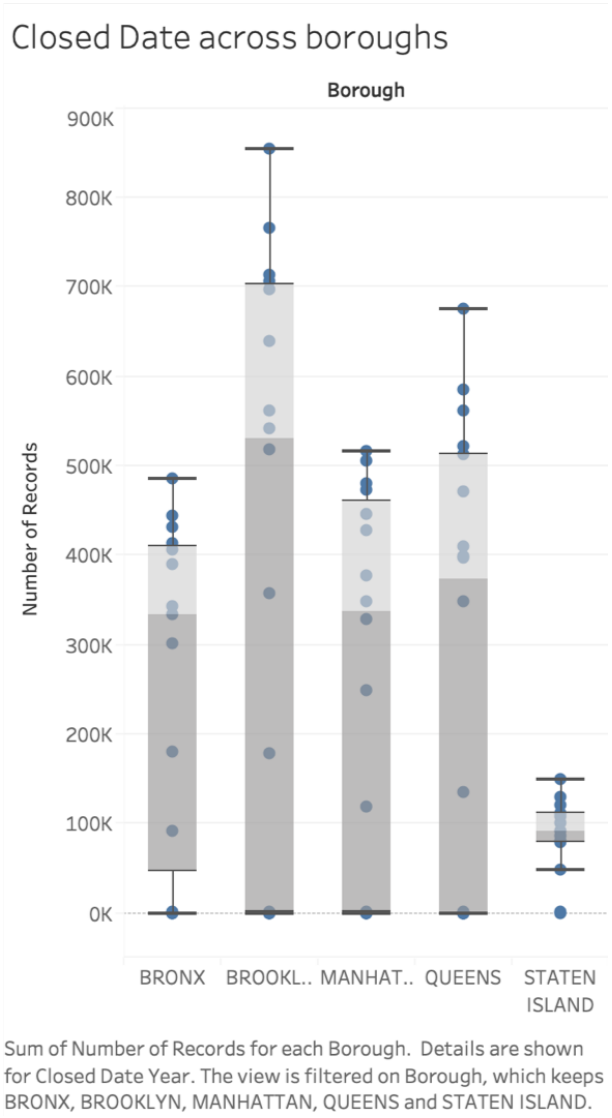

Figure 5: Closed Date across boroughs

Figure 4, 5 and 6 depict the types of complaints across boroughs of New York City. These present the distribution of complaints along with various metrics across boroughs of New York City.

6.2 Figure 4

As the name of the dataset suggests, the main idea is to understand the different types of complaints that prevail in New York City. Therefore, in order to explore and dive into the dataset, we decided to understand the distribution of the various different types of complaints across different boroughs of New York City. So why plot this graph?

This graph is the most fundamental important visualizations of the dataset. As the graph depicts, we have the distribution of complaints across the different boroughs that are represented by

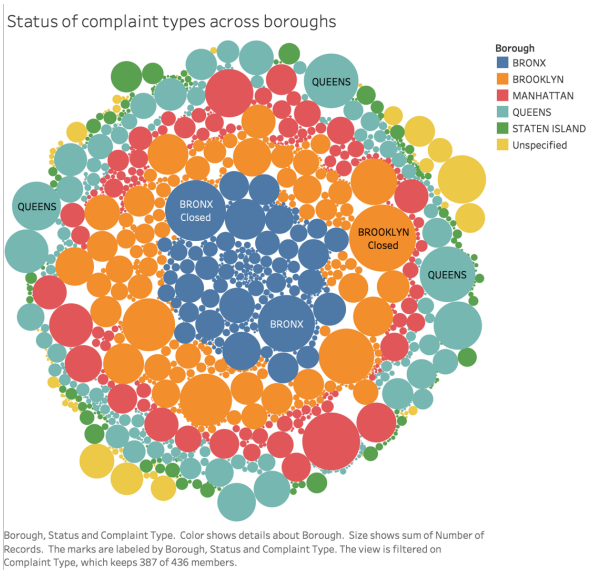

Figure 6: Complaint Type status across boroughs

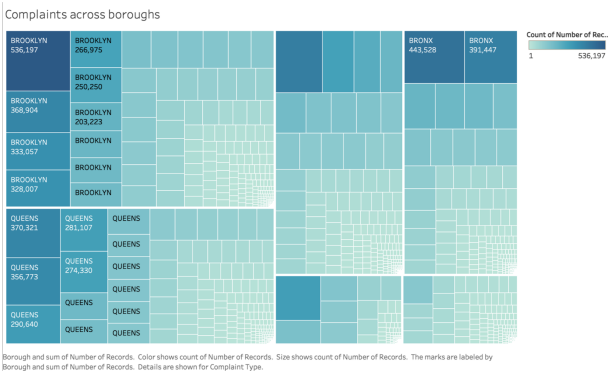

Figure 7: Complaints across boroughs

various colours by the colour scale to the top right corner. The graph tells us about the frequencies of these complaints. From the graph, we can identify the most important complaint types that are currently prevailing in the city, as well as which borough it prevails the most. This plot makes it possible to pick the top 5 most frequent complaint types across different boroughs and dig deeper into them. In the upcoming plots, we will analyse the density of these complaints in different areas of a given borough followed by the location type at which it prevails the most.

6.3 Figure 9

This is one of the most important representations when it comes to attaining a deeper insight about this data. This analysis has shaped direction to the remaining plots. So why is this analysis so important?

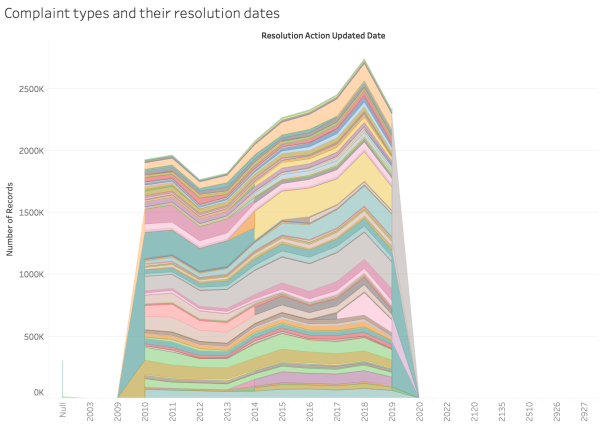

Figure 8: Complaint types over resolution dates

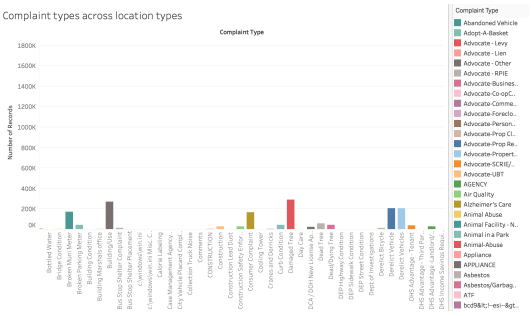

Figure 9: Types of complaints across various location types

One of the main goals of the analysis is to figure out the most common issues faced by people. As the plot depicts, some of the key issues such as air quality, heat issues, damaged trees have a high frequency of complaints raised. This analysis will help us prioritize which complaints need to be addressed and what resources have to be utilized to solve them. Since we have figured out the most common complaint types based on frequency, we can now zero down the boroughs in which these complaints are raised. Furthermore, we can plot heat maps of the complaints for a specific to attain a deeper understanding of which specific regions of the boroughs have to be addressed. Furthermore, we can zero down even more, by plotting these complaint types versus the location types. For example, people living in apartment type houses in queens often complain about heat issues faced during the summer months. This analysis will help the agencies figure out their exact audience and take necessary actions immediately.

6.4 Figure 7

Now that we have figured out the different complaint types and how they are distributed across different boroughs, we need to dive into one of the boroughs and analyse it thoroughly. How do we know which borough to pick?

The plot above answers that question for us. As the plot depicts, we have a heat map of the complaints across the different boroughs of New York City. This plot makes it easier to choose the ordering of the boroughs for analysis. The borough with the most complaints needs the fastest solutions. Hence, from the plotted graph, we can conclude that Brooklyn borough has the most number of complaints and hence we analyse it next.

6.5 Figure 10

Since we analysed the various different complaint types that are present across the five different boroughs, we now perform analysis that is specific to a borough. One of the most important analysis is the number of open complaints. Open complaints help us in analysing the number of complaints that have been registered with respect to various different agencies and issues. The frequency of the complaints in a region will help us infer many things.

As the plot depicts, we have a heat map of the Brooklyn region that depicts the different open complaints across the Brooklyn are and how they vary. The color scale on the right helps us identify the frequency of the open complaints in a given region of the Brooklyn borough.

As we can see, the highest number of open complaints in this region is 3,51,278.

Now arises the question – how does this analysis help?

Firstly, the frequency of open complaints can help us identify the region that needs maximum attention first. This analysis will help the agencies understand which region of Brooklyn they need to address and tackle first.

Secondly, the number of open complaints in a given region can help us zero down to the most common complaint in this specific region. This data can be obtained by searching along these latitude and longitude values. On the contrary, it will also help us to retrieve information about – which agency has received the most number of complaints. Furthermore, this will also help us deduce and conclude about the responsiveness of the different agencies in this region. This can be achieved by analysing the number of registered complaints versus the number of open complaints with respect to a certain agency.

6.6 Figure 8

After understanding the number of complaints across boroughs, we looked at the types of complaints with their dates of resolutions of respective complaints. We achieved this using area charts with years plotted on x-axis and the number of records for each data type plotted on the y-axis. Area for complaint type would help us understand the relation of complaints over resolution dates.

6.7 Figure 5

Along with the number of complaints, we moved on to analyzing the closed date for complaints across boroughs. We used box plot to achieve this to identify the outliers which don't follow the closed dates pattern. Box plot helps us to get median along with quarters for the complaints across all the boroughs.

6.8 Figure 6

After analyzing the relationship between complaint types and resolution dates, we moved on to complaint type status across boroughs. With the use of bubble chart, we were able to represent the number of records for each complaint type across boroughs with size of the bubble as metric to represent this information. This made the visualization intuitive and colorful as we could analyze the complaint types across all the boroughs.

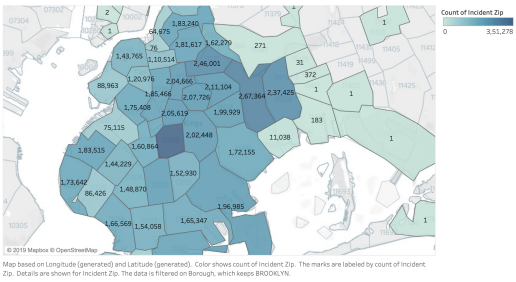

Figure 10: Distribution of incident zip across New York City

7 SUBMISSION

You can find the code on the GitHub repository:
https://github.com/gandalf1819/NYCOpenData-Profiling-Analysis

DUMBO HDFS directory:
/user/cnw282/2019BD-project-results

8 CONTRIBUTIONS

Task-1:
For task 1, we all brainstormed on which framework to use for creating a dataframe for each csv file keeping in mind that the task has to be computed in distributed manner. Initally, Vineet suggested the use of RDDs for the task but we found the use of spark dataframes to be more effective and efficient than RDDs. Hemanth wanted to use Spark SQL queries for operating over dataframes but this had limitation as we had to create views for each of the datatype identifiers. Finally, Chinmay suggested the use of *udf* functions coupled with *collect* method as this would optimize the process of collecting rows for each datatype. This approach turned out be highly efficient and the most distributive method to get the results from all the JSONs. Thus, infrastructure and framworks were finalized for Task-1 and with the distributive power of DUMBO and AWS tx.2xlarge instance, we were able to collect 1790 JSONs.

Task-2: Since, Chinmay had already setup the infrastructure for Task-1, he suggested the used of distributed pandas framework - **Modin** which uses Ray and Dask as distributed infrastructure to speed up the task. This was again the best distributive approach

while processing the NYCColumns dataset files from cluster2.txt. For methodologies, Vineet suggested the use of Regular expressions for columns which followed a standard pattern. This turned out to be particularly useful as such columns were processed very efficiently. In order to improve this and be applicable for remaining columns, Hemanth suggested the use of Fuzzy ratio with Levenshtein distance for columns where we can create a global dictionary and process the rows for all the columns. This approach proved to be the best to process the remaining columns.

Task-3: After finishing the task-3, we collective thought of answering the questions for NYC 311 datasets and make the visualizations more intuitive. We added visualizations like stacked bar charts, bubble chart, tree map and small multiples and area charts to create intuitive visualizations to identify the complaint types in boroughs, status of complaints, open and closed dates of the complaints and so on.

REFERENCES

[1] Yeye He and Dong Xin. SEISA: set expansion by iterative similarity aggregation. International conference on World Wide Web (WWW '11), 2011

[2] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. Table union search on open data. Proc. VLDB Endow. 11, 7 (March 2018), 813-825

[3] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. Automatic discovery of attributes in relational databases. ACM SIGMOD International Conference on Management of data (SIGMOD '11), 2011

[4] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD '19), 2019

[5] Andrew Ilyas, Joana M. F. da Trindade, Raul Castro Fernandez, Samuel Madden. Extracting Syntactical Patterns from Databases. ICDE 2018

[6] Andrew Ilyas, Joana M. F. da Trindade, Raul Castro Fernandez, Samuel Madden. Extracting Syntactical Patterns from Databases. ICDE 2018

[7] Efficient Algorithms for Mining Outliers from Large Data Sets. Ramaswamy et al., SIGMOD 2000

[8] Ming Hua, Jian Pei: Cleaning disguised missing data: a heuristic approach. KDD 2007: 950-958

[9] Sumit Goswami, Mayank Singh Shishodia. A Fuzzy Based Approach To Text Mining And Document Clustering . 2013