Big Data Final Report

Reuben Varghese
Computer Science
New York University
New York
rv1130@nyu.edu

Sai Likhith Kanuparthi
Computer Science
New York University
New York
slk522@nyu.edu

Kartikeya Negi
Computer Science
NewYork University
New York
kn1481@nyu.edu

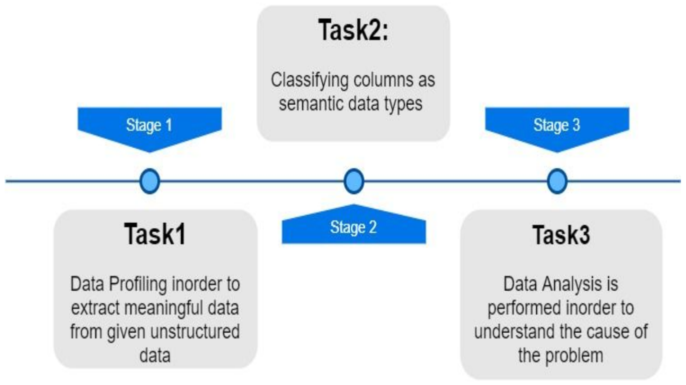
ABSTRACT

Big data has revolutionized how we look at things. Due to its ever-changing nature and the enormous impact it has around us, it is important to understand what big data is, and how it works. The key objective of this work is to try to simulate a set of real-world big data problems and solve them meaningfully. The solutions that we propose are meant to act as examples for solving such problems encountered in any domain, provided there is data involved. Essentially, this body of work strives to solve the general problems that are encountered in data analysis, like generic data profiling, semantic analysis, and data analysis to derive insights from data. For the purpose of simulating a real-world scenario, we apply our solution to large number of datasets taken from NYC Open Data while adhering to strict limitations on the time and resources needed to perform these tasks.

1. Introduction

Data is indispensable in any possible domain that we can think of in today's world. Big Data technologies are on the rise to analyze this data for gaining insights that can help in making strategic decisions. Massive amounts of data are produced every day by businesses, users and other entities, big and small. Big data tasks involve the process of examining the large datasets to generate insights and patterns. Although this field has existed for some time, the new benefits that big data brings to the table are speed and efficiency. The ability to work faster – and stay agile – gives modern technology an edge that it didn't have before. Big Data is indeed a revolution in the fields of Information and Technology.

This project strives to demonstrate a step by step approach that begins with gaining an understanding of the available data. Then, we dive deeper try to gain further knowledge about the semantics, where we deduce the type of real world information the data contains. Finally, we try to derive knowledge that is hidden within the data, something that is not visible at the first glance, but can help answer meaningful questions about the real world. Following this structure, the project is divided into three main tasks: generic data profiling, semantic analysis, and data analysis of the NYC Open Data. We use NYU's Dumbo (high performance computing cluster) infrastructure which provides the computational resources needed to execute big data tasks through parallel processing. Due to certain limitations, we have also used our own Amazon Web Services instances to run each of the tasks.


Task2:
Classifying columns as semantic data types
Stage 1
Stage 3
Task1
Data Profiling inorder to extract meaningful data from given unstructured data
Stage 2
Task3
Data Analysis is performed inorder to understand the cause of the problem

2. Generic Profiling

For our first task, we are provided with a large collection of unstructured datasets from NYC Open Data. These datasets come with very little metadata. Our task involves structure and content discovery, to help a person or machine read, understand and then hopefully analyse our data. In order to get a structure out of our data, we perform data profiling to transform the given datasets meaningfully by deriving metadata that can then be used for data discovery, querying purposes and for identification of data quality issues. For small datasets, data profiling can be done quite faster when compared to the larger datasets, however, for large datasets like NYC Open Data- datasets which contain thousands of records in it, data profiling is quite difficult with generic python. We execute the task using a distributed approach, making use of Hadoop distributed file system (HDFS). This can be achieved using pyspark framework a spark framework for python, since it has parallel processing.

2.1 Methods and Functions used:

The following is a list of methods that were implemented:

count: to obtain the no.of values in a dataset

Identifying missing values: There exist various types of missing values: null, nan, blanks and white spaces etc.; we find the no. of missing values for each database

Min method: Min method is implemented with the intention of finding the minimum value in that column

Max method: It helps to determine the maximum value in that column

Frequent value method: This will help us to understand that which value is repeated often in that column we took it more than 5 times.

Standard Deviation: Standard deviation of the mean of values that are calculated for int type and float type.

2.1.1Algorithm

```
data profiling method(data set):
       read input from the data set

identify data set columns list

for each column in data set:
       identify datatype of the column
       Based on the data type:
       find no. of non-empty cells
       find no. of empty cells
       find the minimum value
       find the maximum value
       find the frequent value
       find the mean of the values
       find the std deviation of values in
       the column

for each data set:
       call the method store the output in
       separate json file

merge the outputs to a json file named
task1.json
```

2.1.2 Innovation:

In the code which we had written for the algorithm, we have used the code in such a way that it takes only datasets.tsv to our local repository and using its index, we generated output json files from the HDFS repository.

Fuzzy logics are used in order to find the proper date format despite the data being unstructured.

Lambda functions and Resilient Distributed Datasets are used for efficient memory usage and to improve execution time.

2.1.3 Challenges

Runtime constraint:
Initially, we implemented data queries with sql. For larger datasets such as ours, using spark sql was inefficient and taking a long time to run, so we used RDDs instead for faster processing.

Memory constraint:
For a few functions which required storing a huge buffer, a lot of storage space was being occupied during the time of execution. So, we implemented Python's lambda functions instead for efficient operations.

Execution memory limit:
Since, we were executing the program on the Dumbo cluster, we were limited by the executable memory. This problem was solved by manually increasing the executable memory while submitting the query for it to run without interruption. The shell command for the same is:

```
spark-submit --master yarn --num-executors 6
--driver-memory 16g --executor-memory 7g
filename.py
```

Excessive Data Types:
Much of the data was represented in the form of the string data type. We had to typecast this data to the relevant data types.

2.2 Evaluation

We aggregated the four data types for all the 1901 datasets and plotted our observations on a histogram for visual understanding (fig 2.1). The data types we dealt with were:

TEXT: This is the most general form of data, Sometimes, the null values were initially classified as string elements for an unstructured dataset.

INTEGER(LONG): The integer type was a highly right-skewed distribution. It was obvious that most datasets did not have many columns that are of integer type. Most of the datasets had less than fifteen columns of the integer type.

REAL NUMBERS: This was a right-skewed distribution as well. There were a few outlying datasets which had more than 20 columns of this type.

DATE/TIME: This data type was slightly similar to real type distribution. There were some datasets that had more than 20 columns of this type.

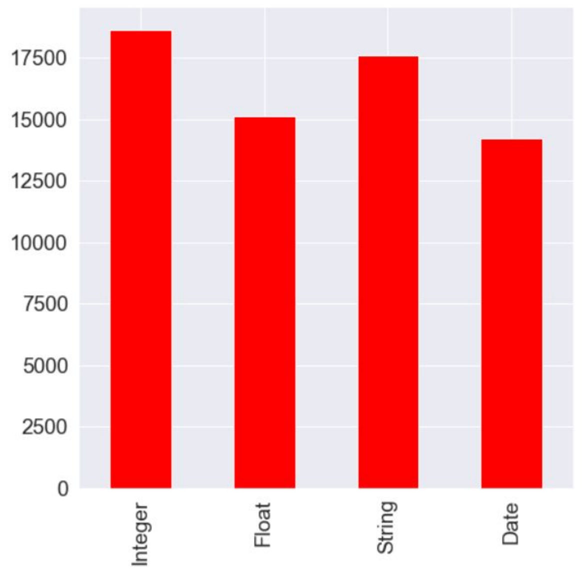


Figure 2.1: Different data types against their count

The bar graph shown above helps us visualize the different data types against their frequency in the output. This in turn helps us identify the dominance of one datatype over the others. In this case, it's integer data type values are more in number when compared to other data types. If we can understand the dominant data type, we get a good sense of what the essence of the data is.

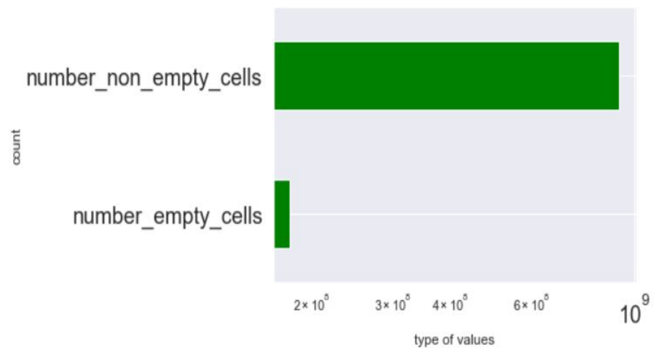


Figure 2.2: number of empty vs nonempty cells

The above bar graph gives us the number for the empty cells vs the number of non empty cells. This analysis helps us to determine the percentage of the unstructured data in our datasets and thus helps us setup guiding points for data profiling. In our case, the non-empty values are more in number when compared to empty values which makes data profiling a bit easier. This metric also acts as an indicator of data quality.

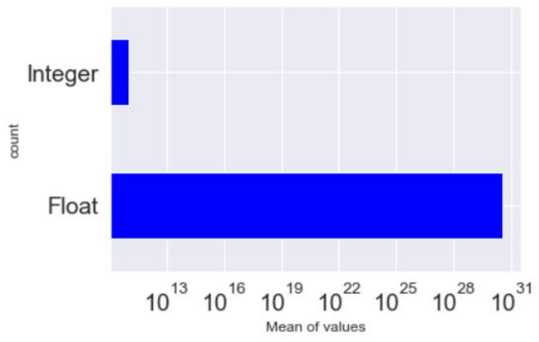


Figure 2.3: Different data types against their mean

The above bar graph is going to give us the visualization for the different data types against their mean values in the output. This helps us to understand the contribution of a particular data type to the overall value., i.e, the data type with higher mean indicates that major transactions are done in the given data type and we can predict the behaviour of applications that have such data.

3. Semantic Profiling

In the previous section we got an idea of the kind of data we are dealing with. From our analysis of the results, we were able to answer questions like what data types are present in each column, how sparse the data is, the mean of the values, etc. among other useful information. This analysis has equipped us for the next task, which deals with the semantic classification of columns. Semantic analysis in our case refers to deducing the type of real world information that is being depicted in a column.

For instance, let us consider a sample column dataset "3rfa-3xsf".Cross_Street_2". We know that this particular column contains a list of streets in New York City by looking at its title and the values inside it. However, if we are to automate this process, we need to come up with ways to make a system understand what kind of real world entity is being depicted by an instance in the column. Also, the data in our columns is of the data-type type 'string', which makes the use of data type as a factor in identifying the semantic type void. There isn't a one size fit all solution to this problem. Note that we are dealing with 23 semantic types like person name, vehicle model make, telephones, etc.

3.1 Methods

The first step is to make an indicator function for each of these semantic types. One of the trivial approaches that we follow for semantic type detection are the use of regular expressions, and fuzzy logic. These work well for those data types that need an ontology or that can be found using regular expressions. One such example would be detecting if a value is a phone number.

3.1.1 Regular Expressions

For some semantic types like phone number and geographical location (latitude and longitude) which adhere to a standard format, we match a column entry with a standard regular expressions in the indicator function for that semantic type. An example of this is `((\(\d{3}\) ?)|(\d{3}-))?\d{3}-\d{4}` which is the regular expression for phone numbers as observed in our data.

3.1.2 Fuzzy string matching

For other semantic types like building type, parks, and vehicle make, we use the fuzzy logic (python's fuzzywuzzy package) where we make a primary list of possible words/sub-strings that such types may contain. Then, we 'fuzzify' our column instance and check if this closely matches with any of the items in the list. We set the fuzzy string similarity threshold to a custom value. Thus, if two strings have a fuzzy similarity equal to or more than the given threshold value, the output is a match. The lists for these tasks have been built from various sources. A few of the sources are mentioned below:

Semantic Type	Source
Vehicle Types	New York DMV[5]
Building Classification	NYC Official Website[6]
Park/PlayGround	Wikipedia[7]
NYC Agencies List	NYC Resources Website[8]

Table 1: List of sources for various ontologies used

3.1.3 Name Entity Recognition

This is a method in which a given text is chunked into different entities. One way of implementing name entity recognition is by training a deep learning model on a given dataset.This task can also be performed by simply matching with an existing ontology but for entities such as person name, these ontologies can be exhaustive and hence we need a better approach to handle such tasks. Stanford has a library called the name-entry-recognition library that uses a natural language processing model to classify input into one of many predefined semantic types. This was particularly helpful in classifying entries as person/business name semantic types which do not have a well defined rule/structure. Currently, the stanford ner library provides us with only a maximum of 7 entities that it can recognize namely, location, person name, business name, date, money, percent and time.

3.2 Challenges

3.2.1 One to many mapping

Lots of column entries can be classified as a particular data type. In other words, an entry itself does not give us enough information to satisfactorily classify it into one of the semantic types. An example is a street, say '21st St' which would satisfy the conditions for it to be classified as both an address and as a street name. This error can however be resolved by taking into consideration the classifications of its neighbors in the same column along with the column name. The column name gives us an idea of what semantic type to expect in that column as in most cases, it contains the types of data found inside it. Thus, we implemented fuzzy logic on the name of the column file to first find the most likely semantic type values it contains, and then to apply that function to the entries before the other functions. This proved to be a good approach as it reduced our task of manually labeling the ground truth of each column, and helped us use a previously implemented function on metadata as a precursor to the actual task.

Now, one of the ways of tackling this issue is to use fuzzy logic with a threshold. This will ensure that not every function returns a positive output for any given input, Another factor to consider would be that for ontologies with smaller values in them such as DMV types which consist of small codes, it is very easy to match them with almost any input since the ratio is high most of the time. To prevent this one to many issues from occurring, we try to guess the type of a column from the column name.Using this information, we already have a hunch of what semantic type most of the values in this column may belong to.

3.2.2 Arbitrariness of semantics

For semantic types like person name (first name, last name, etc.) and business names, it was difficult to formulate a rule/algorithm that was able to fully capture the logic behind that semantic type. For example, if we implement a fuzzy logic using lists of generic words for a person’s name, we will have to create a very large list that has all the possible names that people can have. This would lead to an accurate but a very computationally expensive search for each entry in every column. Keeping in mind the time needed to execute such a task, this idea was discarded immediately. Instead, the process of first finding the most probable semantic type using fuzzy logic on column name was implemented and found to be useful again. An alternate approach that worked one these semantic types was using Stanford’s named entity recognition library. This library uses Natural language processing based prediction models that do a better job at recognizing person names but at the cost of time.

3.2.3 Missing Values

Many columns had missing/null values. These values could be represented in any value from being described as null to being empty strings. Thinking of a good way to deal with them is vital in helping us get meaningful results.

3.2.4 Runtime

Due to resource and time constraints, we had to come up with an optimized algorithm that could greatly reduce the amount of time spent executing each task. The brute force approach in which we check for the type by iterating through all the methods we have gave us a very poor runtime. The larger datasets took almost two to three hours to analyze.
One vital piece of information that can be used to greatly optimize the code is the column name. In most cases, the column name was closely in tandem with that the column values represent. Hence, using the column name, instead of blindly iterating through all the functions created, we directly first check to see if the value is of the same semantic type as depicted by the column name. In most cases, this approach directly eliminates the need to iterate through any of the other functions. This approach also ensures a higher accuracy given that the probability of getting is higher the more functions we run.

We also trivially improved the accuracy by ensuring that functions return the moment they become positive for a particular input.

3.3 Evaluation

In this task, we designed a code that would predict the column types of about 272 columns. For each of these columns, we plotted a graph showing us the 5 most common semantic types that may be present in that column.

For each column, we also generate a json file encapsulating the same information.

On our first iteration, we were getting a much lower accuracy owing to the fact that the data is arbitrary and because the generated output can trigger a positive outcome from multiple functions due to the close similarity in our given data, For example, “Broadway” can be both a street and a landmark or business name.

The following shows the output on a sample number of datasets for top 5 number of columns matched.This graph can help us gain insight into what semantic types are being detected most accurately.

So, as we can infer from Fig.3.1, street name matched well followed by subjects in school which is followed by school level, borough and neighborhood.

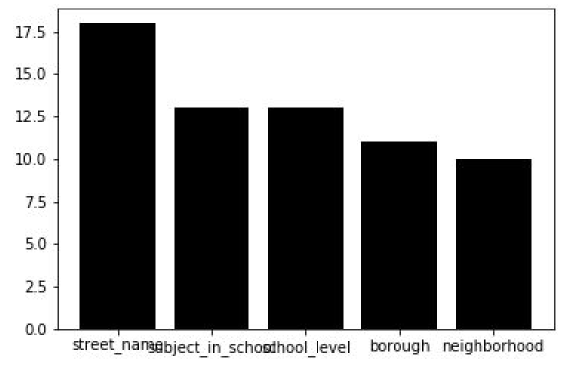


Fig 3.1 Top 5 columns that matched the most

Similarly, Fig 3.2 contains a plot of a column containing multiple semantic types..

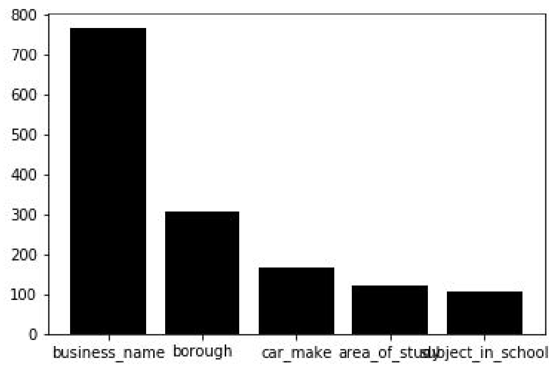


Fig 3.2 Example of column with multiple semantic types

4. Data analysis

Data analysis is the final and arguably the most important part of big data. It gives us an understanding of the real world from the data. In some ways, it models the world represented by our data and lets us explore that world and gain valuable information in the process.

For our analysis we have used the NYC 311 dataset that contains a list of non emergency complaints logs. We analyse only the month of November as we want to analyse the most recent and relevant data. We also want to analyse what kind of an impact a holiday like Thanksgiving (November 28) has on our analysis.

In this section, we answer very specific questions pertaining to gaining an understanding of the world modelled by data. These questions are: What are the three most frequent 311 complaint types by borough. 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?

The image below informs us about the top three types of complaints logged for each borough. It plots the complaints against their normalized frequency (out of 1).

4.1 Inferences:

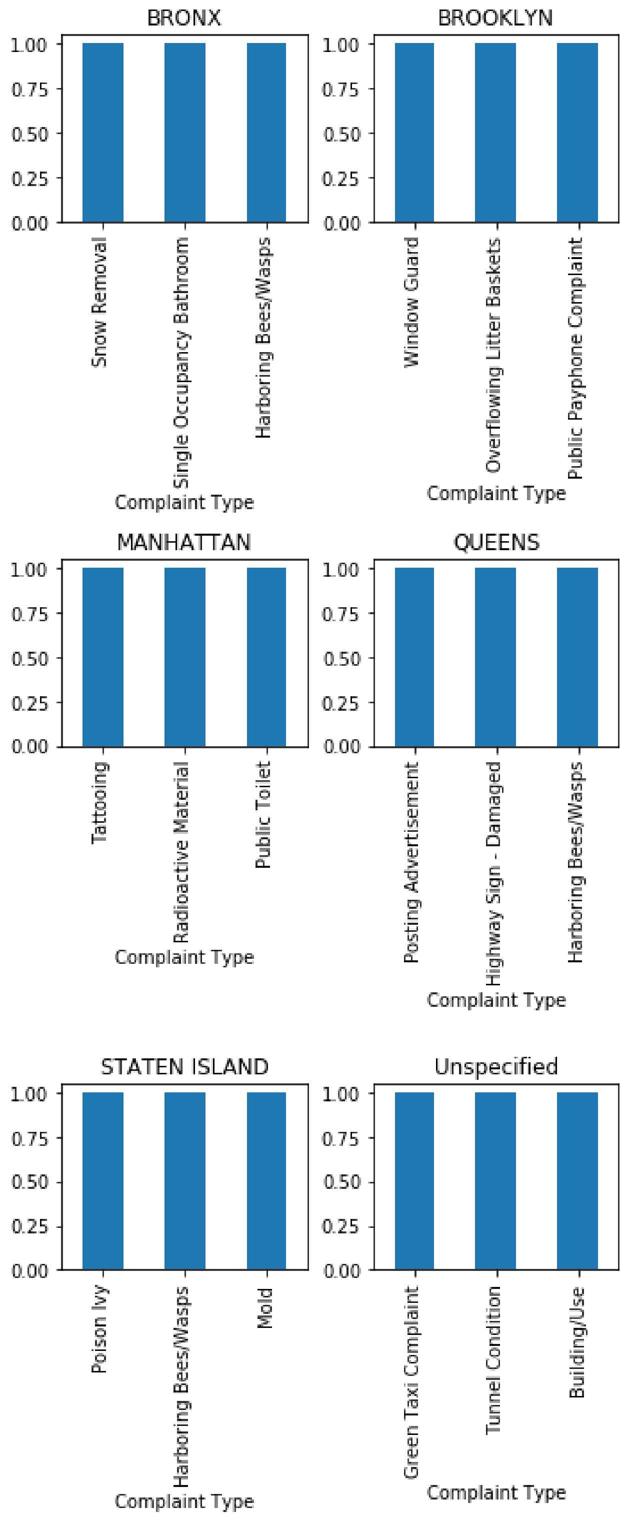
As we can observe, the top three complaints for each of the boroughs appear to be equally frequent. There is not much difference in the number of occurrences of these complaints. This allows us to infer that there is no particularly frequent and alarming issue that is inherent to NYC.

Moving forward, we can see that none of the frequent complaints are repeated among the five boroughs. This leads us to the conclusion that each of the boroughs has its own problems inherent to its location, demographics, economics, etc.

For example, Brooklyn, being the most populous neighborhood of New York City, will have a higher number of complaints related to overflowing litter baskets due to more waste being generated..

An example pertaining to Manhattan shows that there are lots of complaints against public toilets. This can be explained by the fact that being overcrowded with commercial buildings and retail stores, Manhattan has very few public toilets available. This factor, combined with the huge inflow of tourists every day leads to shabby condition of the few toilets that are open to the public.

Considering the large number of green taxi complaints that are not classified into either of the boroughs, the best possible explanation for this phenomenon can be that taxi trips usually happen between boroughs and hence, it would not make sense to give a complaint against green taxis while mentioning the location/borough for that complaint. Also, since green taxis have been introduced recently and there may be problems pertaining to their frequence, service, the areas they cater to, etc. leading to a large number of complaints against them.



We have left the last part pertaining to the change in complaint frequencies unanswered as it is evident that during the November 26 to November 30 holiday week, one would expect more complaints related to garbage and waste disposal due to higher quantities of food waste being generated during this time.

INDIVIDUAL CONTRIBUTION

Kartikeya Negi (kn1481)

Kartikeya contributed towards finding solutions to tasks 1 and 2 along with the other teammates, visualized the results, and wrote the output for task 3. He also contributed in writing the project report.

Sai Likhith (slk522)

Sai Likhith contributed mainly towards writing the code for Task 1 and generating the results. He was also involved in brainstorming and developing techniques to solve the issues in task 2 and 3. He also contributed in writing the project report.

Reuben Varghese (rv1130)

Reuben contributed mainly towards developing and implementing task 2 and generating the output. He was also involved in developing techniques to implement all the tasks as well.He also contributed in writing the project report and in creating the Github repository.

5. Conclusion

This body of work effectively performed the tasks of data profiling and data analysis on the NYU Open Data. We started off with the generic profiling of datasets where we identified the various data types present in our data. Then, we extracted the semantics of column data provided to us using both home grown and well known techniques to generate an algorithm that gave high precision and recall while classifying data as the various semantic types. Afterward, we moved on to the important task of data analysis where we visualised and extracted information from the NYU 311 data. Then, we applied reasoning to validate our results. Overall, the essence of a real world big-data problem was successfully captured and solved in this project, giving us experience to deal with such problems later in life.

ACKNOWLEDGMENTS

We thank Professor Juliana Freire and the guest lecturers for teaching the material on big data analysis, for answering our questions and for helping us develop an immense interest in big data. They equipped us with the necessary skills and knowledge to finish a project of this magnitude and complexity. We also are extremely thankful to the teaching assistants for their contribution towards strengthening our conceptual and technical understanding of the various topics in big data.

REFERENCES

[1] How to find nan and null values of a column using pandas-https://stackoverflow.com/questions/44627386/how-to-find-count-of-null-and-nan-values-for-each-column-in-a-pyspark-dataframe
[2] Pandas dataframe plot examples with matlib pyplot-http://queirozf.com/entries/pandas-dataframe-plot-examples-with-matplotlib-pyplot
[3] What is data profiling and how does it make big data easier:https://www.sas.com/en_us/insights/articles/data-management/what-is-data-profiling-and-how-does-it-make-big-data-easier.html#/
[4] Pandas Dataframe plot examples with matplot lib-http://queirozf.com/entries/pandas-dataframe-plot-examples-with-matplotlib-pyplot#plot-histogram-of-column-values
[5] Potoski, Luke. "Registration Vehicle Types." New York DMV,6Feb.2019,dmv.ny.gov/registration/registration-vehicle-types. Accessed 11 Dec. 2019.
[6] "NYC Building Classifications - DOF." Nyc.Gov, 2019, www1.nyc.gov/assets/finance/jump/hlpbldgcode.html. Accessed 11 Dec. 2019.
[7] Wikipedia Contributors. "List of New York City Parks." Wikipedia, Wikimedia Foundation, 10 July 2019, en.wikipedia.org/wiki/List_of_New_York_City_parks

[8] "NYC Resources | Agencies | City of New York." Nyc.Gov, 2019, www1.nyc.gov/nyc-resources/agencies.page. Accessed 11 Dec. 2019
[9] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4 (CONLL '03), Vol. 4. Association for Computational Linguistics, Stroudsburg, PA, USA, 180-183.
[10] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çağatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330993

Link to github repository:

https://github.com/reubenvarghese1/CSGY-6513-Big-Data-Project-Analysis-of-NYC-Open-Data