# Big Data Final Project Report

Yijun Hu
yh2666@nyu.edu
New York University, Tandon School of Engineering
New York

Yi-Hsuan Wang
yhw288@nyu.edu
New York University, Tandon School of Engineering
New York

Tianming Wang
tw1838@nyu.edu
New York University, Tandon School of Engineering
New York

## ABSTRACT

This is the report written in LaTeXAnd it aims to present what we have done in the project of CS-GY 6513 Big Data for 2019 New York University, Tandon School of Engineering. The purpose of our project is to profile NYC Open Data using Python, Spark, SQL, etc. and visualize the results using DataWrapper. The project is divided into 3 parts. In the first two parts, we try to profile all the datasets with both manual recognition and Spark pipeline. In the third part, we pick a specific dataset and dig into it, bring out a visualization result based on an analysis by answering the question.

## KEYWORDS

Big data, NYC open data

**ACM Reference Format:**
Yijun Hu, Yi-Hsuan Wang, and Tianming Wang. 2018. Big Data Final Project Report. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

We have designed and implemented methods to handle big data infrastructure. In this project, we are handling around 1900 datasets from NYC open data, and the largest one is up to 3 GB.

In task1, we focus on Generic Profiling. We are profiling a large collection of open data sets and derive metadata that can be used for data discovery, querying, and identification of data quality problems.

For task2, we are doing Semantic Profiling. First, we label each column with True values, and then we derived methods to add one or more semantic type labels to the column metadata.

For task3, we focus on the "NYC 311 service dataset from 2010 to present" table. Our goal is to find how fast does NYC address curb and sidewalk service requests? And are these requests addressed more promptly in some zip codes than in others?


*Woodstock '18, June 03–05, 2018, Woodstock, NY*

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

## 2 TASK 1

### 2.1 Method

Our idea is that using Python's own type conversion function to try to convert a tuple to a certain type. If it returns false, try another type; Because of the problem of various types, which will be explained detailed in the challenge section, the order of the type-detecting is specified: integer, real, and date/time, after trying all the order of detection.

### 2.2 Challenge and Time

we each spend 3 days to run all the data and faces challenges including dumbo volume limitation, data quality, and classification issue. As data quality and optimization to solve volume limitation will be discussed in 1.3, 1.4, respectfully, here we will only cover classification problem and our compromise due to the large volume in the Dumbo platform.

Various types of a cell: one tuple can be labeled as many types, e.g. every tuple with integer label can also be labeled as float, and every tuple can be labeled as text. But a tuple with multiple types is of no use to our analysis and may cause problems. So, we choose to assign one type to one tuple and rearrange the decision statement to classify the tuple as the most relevant label.

Due to volume limitations, we cannot classify all the data sets. After using optimizations, we still find it is slow to run all the data, for some of the datasets have hundreds of millions of columns, e.g. one file has been run for 12 hours without completing. As we cannot keep the program on 24 hours a day, we drop some of the data sets.

### 2.3 Optimizations

Sorting file: as the program will be stuck in big files, we sort all the files by their size, and run small files first.

One file one output: We encounter several crashes when trying to run the whole data sets in a single program, which helps to output counts and co-occur type in a column. So we decide to create a JSON file per data set first and to merge them into one file to get statistical results later. As other teams struggle with crashes, our team has all the data already.

Cache: As there will be many aggregation operations on the same table, and spark is lazy, we save those results in the cache first and then read them from memory directly.

### 2.4 Data quality issues

(1) too many NULL values: when we start to filter the data, we find that there are lots of NULL values and empty cells in each column, which will be an obstacle in the statistical analysis.

(2) heterogeneous columns: most columns contain only one type of data, however, some may not be the same, e.g. one column labeled as zip code can contain integer-type data, e.g. 10950, and string-type data NY 10950.
(3) Various representations: e.g. the date type has too many data types, except for forms which parse() from python.datautil package can handle. For instance, YYYYMM, MMDD, YYYY.MM, MM.DD(Y refers year, M refers month, and D refers day).

## 2.5 Analysis

**Summary**: Figure 1 shows for each data type, how many columns contain that type. And we can see from the graph that most columns are of integer type, second-most columns are of text type, third-most are real and last are date or time.

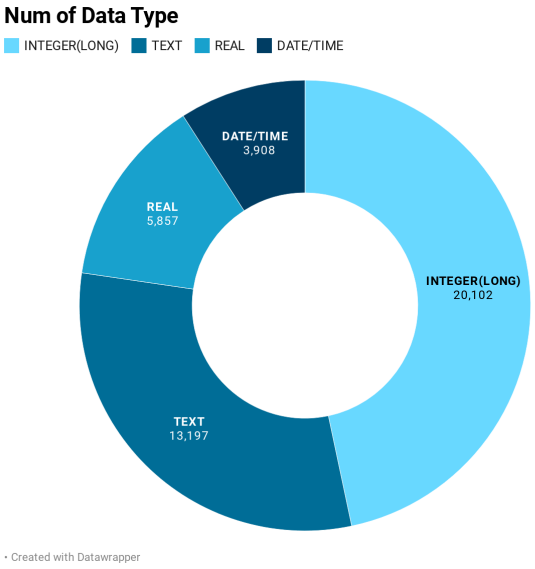


**Figure 1: It shows for each data type, how many columns contain that type.**

**Most common type**: we can conclude most common types that co-occur in a column.

Figure 2 shows the frequency of different itemsets. we can conclude that for 2-frequent itemsets, integer(long) and real co-occur most, followed by integer and date/time; And for 3-frequent itemsets, integer, real, and date/time co-occur most.

# 3 TASK 2

## 3.1 Strategy

We tried 2 strategies to detect the different types, feature extraction, and k-means.

## 3.2 Plan A: feature extraction

Here are the steps of plan A:

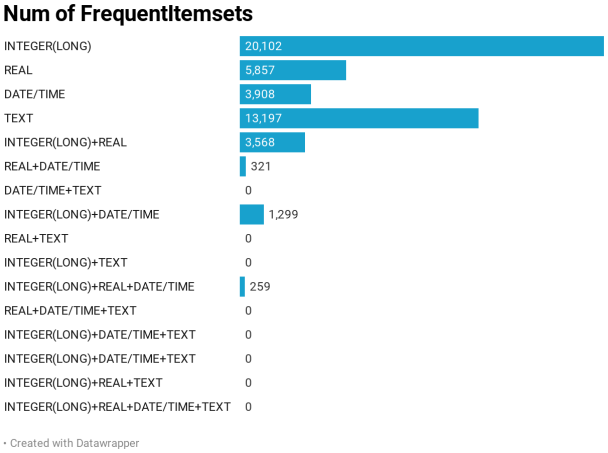


**Figure 2: It shows shows the frequency of different itemsets.**

(1) use column name to determine the range of data type. e.g. if the column name is "Name", it can only be labeled as "person name", "business name", "school name", and "street name".
(2) Use regular expression and datasets to verify the exact type of each tuple. Use regular expression to verify data with a specific format. Use out-sourced sets as master data to check if the tuple belongs to the set.

**Benefits:**

(1) After finding all the master data sets, it quite straight-forward to identify all the columns without training the data.
(2) It takes less time to run the program.

**Limitations:**

(1) It can only be applied in these particular datasets, and if we want to do semantic analysis to other datasets with other labels, we need to extract features of new labels again.
(2) It takes time to extract features of each label, and it will be complicated if there are more labels.

**Analysis:** Table 1 is the output of plan A.
Figure 3 is the count of columns of true labels:
Figure 4 shows the count of prediction.
Figure 5 shows the count of columns of correct predictions.

## 3.3 Plan B: k-means

Here are the steps of plan B:

(1) Split word to tokens(RegexTokenizer), and convert tokens to words vectors(Word2Vec).
(2) Use labeled datasets as train sets to produce clusters.
(3) Use K-means clustering to cluster test words vectors.

**Benefits:**

(1) We do not have to identify the features of each label.
(2) The train datasets can be used as train datasets.
(3) the program will be portable if the size of datasets or labels grows.

**Limitations:**

**Table 1: Precision or Recall of each Label**

| Label | Precision | Recall |
|---|---|---|
| person-name | 0.22222 | 0.07692 |
| business-name | 0.7500 | 0.35294 |
| phone-number | 1.00000 | 1.00000 |
| address | 0.78571 | 0.68750 |
| street-name | 0.81818 | 0.81818 |
| city | 0.68421 | 1.00000 |
| neighborhood | 1.0000 | 0.04545 |
| lat-lon-cord | 1.00000 | 1.00000 |
| zip-code | 1.00000 | 1.00000 |
| borough | 0.56757 | 0.95455 |
| school-name | 0.71429 | 0.93750 |
| color | 1.00000 | 1.00000 |
| car-make | 0.57692 | 1.00000 |
| city-agency | 0.92308 | 0.92308 |
| area-of-study | 0.44000 | 1.00000 |
| subject-in-school | 1.00000 | 0.15789 |
| school-level | 0.51852 | 1.00000 |
| college-name | 0.50000 | 0.40000 |
| website | 1.00000 | 0.77778 |
| building-classification | 0.90000 | 1.00000 |
| vehicle-type | 0.48000 | 0.92308 |
| location-type | 1.00000 | 0.50000 |
| park-playground | 0.66667 | 0.66667 |
| other | 0.12037 | 0.76471 |

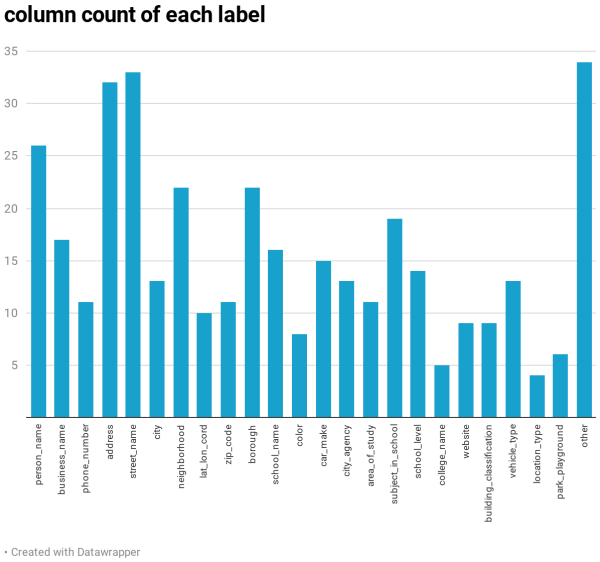


**Figure 3: It shows the count of columns of true labels**

(1) It takes a very long time to train the data, so we only train the cluster with limited train sets.

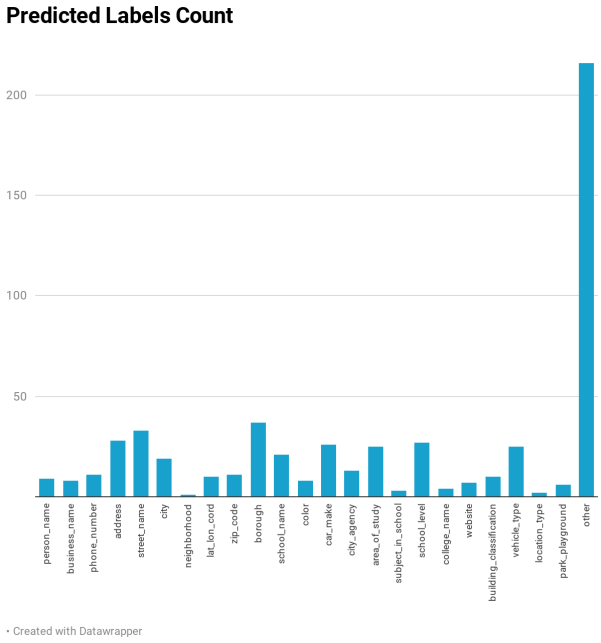


**Figure 4: It shows the count of columns of prediction labels:**

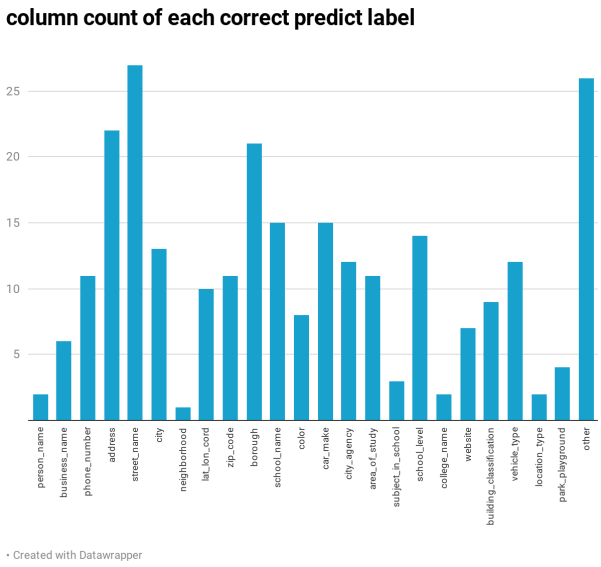


**Figure 5: It shows the count of columns of prediction labels:**

(2) K- means is an unsupervised way to cluster, so results may not be clustered as we labeled before.
(3) The clustering result may not have any semantic meanings, as they only have numbered labels.
(4) The results should be revised manually.

**Experiments Result:**

(1) Dumbo reaches 2.5 GB memory overflow when we try to use every table as the training data, so we take the smallest dataset in 23 labels instead.
(2) There are no weighted-K-means in PySpark lib, so we transformed data into RDD and flat them into the correct amount.
(3) For those data with numbers, ex: lat/lon, zip, and phonenum, are clustered into same label.
(4) For those data with "street" and "avenue", the model can cluster them well.
(5) Our model can correctly predict "park" and "school" into different clustered.
(6) For words in Type of Location, word2vec can't transform the semantic meanings of words, ex: Bank and Airport may not have a similar vector after transformed.
(7) As we only use the smallest dataset of training date, we have bad reulst as Figure 6 shows, so we decide discard this plan. And there are no outputs.

```
|           N MERRICK|        4|
|              PIER A|        4|
|POWELLS COVE BOUL...|        4|
|PRE EASTBOUND ENT...|        4|
|PREK 1423 62ND ST...|        4|
|PRE_K  ROOSEVELT ...|        4|
|PS 93 AX ECFBD ST...|        4|
|PSIS 189 STEENWIC...|        4|
|QN MDTWN TNNL APP...|        4|
|QUEENSBORO BRDG EXIT|        4|
|RED HOOK NEIGHBOR...|        4|
|REINHARD E KALTEN...|        4|
|     RIDGE BOULEVARD|        4|
| RIP NORTHBOUND EXIT|        4|
|ROSE FEISS BOULEVARD|        4|
|       SHOW THE GOOD|        4|
|SITTING AREA ON 1...|        4|
|THE ACADAMY OF TA...|        4|
|THE BROWNSTONEHAR...|        4|
|      THE CAVU GROUP|        4|
|    THE COPPOLA FIRM|        4|
|    THE CULTURE TREE|        4|
|THE FULL CIRCLE G...|        4|
|THE GLOBAL LEARNI...|        4|
|THE GRANT ACCESS LLC|        4|
|    THE GREENE GRAPE|        4|
|   THE HARQUIN GROUP|        4|
| THE JAY CORPORATION|        4|
|THE LIFESTYLE AGENCY|        4|
|     THE LOCKER LADY|        4|
|THE METROPOLITAN ...|        4|
|THE MONROE GROUP LLC|        4|
|THE MUSIC ROOM AT...|        4|
|THE NICHOLSON COR...|        4|
|   THE PERFECT PROMO|        4|
|      THE PRIME  LLC|        4|
|  THE SIGN WORKS INC|        4|
|     THE SOLIS GROUP|        4|
|     THE TABLE GROUP|        4|
|  THE VON AGENCY INC|        4|
|THURGOOD MARSHALL...|        4|
|                  TO|        4|
|                 VAN|        4|
|                 VAN|        4|
|VERA SECURITY  TR...|        4|
|WILLETS POINT BOU...|        4|
```

**Figure 6: It shows the count of columns of prediction labels:**

# 4 TASK 3

## 4.1 Problem definition

How quickly do NYC address curb and sidewalk service requests? Are these requests addressed more promptly in some zip codes than in others?

## 4.2 Methods

**Data:**we use the "erm2-nwe9" file, which is "311 Service Requests from 2010 to Present".

**Steps:**

(1) look at columns of Descriptor, Location Type and Complaint Type. Select those columns with "Sidewalk" or "Curb"
(2) Subtract "Closed Date" and "Created Date" to get the diff of timestamp
(3) Group those data by "Incident Zip" to compute the average response time of 311 services.

**Limitations:**

(1) It cost around 5 hours to run on dumbo due to the large volume of data.
(2) There are many data that are not in NYC, and also there are some data that has "Closed Date" prior to "Created Date".
(3) We simply handle those problems by recognizing them as outliers and just delete them when doing data analyzing.

## 4.3 Findings

Figure 7 is the graph of output, darker color shows the places where it takes more time to response the service request.

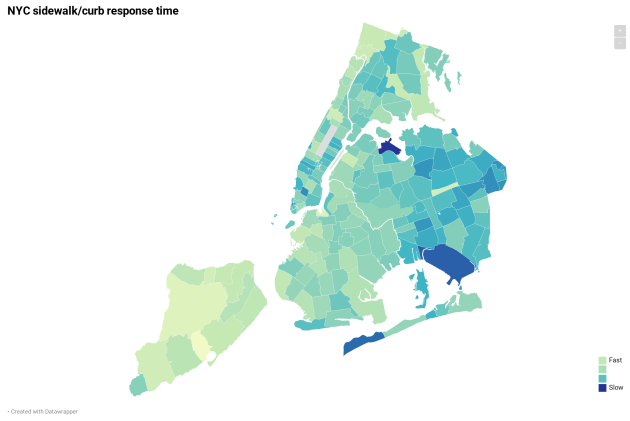


**Figure 7: It shows the count of columns of prediction labels**

**How quickly do NYC address curb and sidewalk service requests?**

(1) On average, the 311 curbs and sidewalk service cost 19.54 days to fulfill.
(2) In most districts, it takes about 15 days to close the request.
(3) There are five districts where it takes more than 100 days, which are Mineola 11501, Manhasset 11031, Yonkers 10703, Manhasset 11020, and Great Neck 11023.

**Are these requests addressed more promptly in some zip codes than in others?**

(1) From the graph, we can see that the average response time in Queens is longer than other districts in Manhattan, Brooklyn, and the Bronx.
(2) There are about 30 districts that spend less than 3 days for requests on average, most of them are in Manhattan.
(3) The longest response time in Manhattan is in 10055, which takes around 90 days.