Data Cleaning at Scale of 311 Service Request from 2010 to present

Ritik Lnu
rl4017@nyu.edu
New York University

Neel Shah
nks8839@nyu.edu
New York University

Vishal Shah
vs2530@nyu.edu
New York University

NYC 311's mission is to provide the public with quick, easy access to all New York City government services and information while offering the best customer service. It helps Agencies improve service delivery by allowing them to focus on their core missions and manage their workload efficiently.In this project we are cleaning the NYC's 311 service Request from 2010 to Present to make the analysis and other visualization easy and accurate.

1 Introduction

Inconsistent, incorrect, and missing data is quite common in today's world which is often unavoidable. It may be due to human errors, inconsistent data management systems, or some other factors. In such cases, good data profiling and data cleaning techniques are quite important and inevitable to obtain data that can be used for further processing. We use data profiling to analyze the structure of our dataset and check if it has a standard structure for further processing.
After evaluating the structure of our dataset we employ several techniques to clean the data. Some techniques are used to remove the outliers and other techniques are used to remove the missing values.
We use some open-source libraries to perform data profiling and data cleansing. We also use some custom User Defined Functions (UDFs) to perform data cleaning which we found to be more effective than the open-source libraries.
We use the 311 Service Requests dataset which encompasses all non-emergency requests from the city, including but not limited to noise complaints, air quality issues and reports of unsanitary conditions, etc. The 311 calls in New York City are publicly available on NYC OpenData.

2 Problem Formulation

Through our data profiling and cleaning of the dataset, we intend to answer the following questions:

1. How can we effectively clean the dataset by focusing on each attribute?
2. How can we sample our dataset so that it resembles our full dataset?
3. How can we use the open-source libraries to profile and clean our dataset?
4. How can we measure the effectiveness of the cleaning techniques?
5. How can we improve our strategies for measuring the cleaning techniques?
6. How can we create the reference data which can be used for future cleaning tasks on other datasets?
7. How can we find more datasets with similar attributes to our selected dataset?

3 Related Work

3.1 NYC Opendata

New York City's open data legislation establishes a comprehensive citywide policy - a shared set of standards and guidelines for the city's ongoing open government efforts – and establishes the Open Data Portal as a centralized repository for the city's open data. Over 1,200 data sets are available through NYC OpenData. Cultural affairs, education, health, housing, property, public safety, social services, transportation, and other City functions are all covered by the data. Other projects, such as the NYC BigApps competition and the work of the Mayor's Office of Data Analytics, are powered by these data, and they open the way for new initiatives to leverage technology and data to engage the public, drive decision-making, and improve government effectiveness.

3.2 Apache Spark

Apache Spark is a distributed processing solution for big data workloads that is open-source. For quick analytic queries against any size of data, it uses in-memory caching and efficient query execution. It offers code reuse across many workloads—batch processing, interactive queries, real-time analytics, machine learning, and graph processing—and provides development APIs in Java, Scala, Python, and R. It's used by companies across the board, including FINRA, Yelp, Zillow, DataXu, the Urban Institute, and CrowdStrike. With 365,000 meetup members in 2017, Apache Spark has become one of the most popular big data distributed processing frameworks.

Hadoop MapReduce is a programming technique that uses a parallel, distributed method to process large data collections. Developers don't have to worry about job distribution or fault tolerance when writing massively parallelized operators. The sequential multi-step process required to run a job, however, is a difficulty for MapReduce. MapReduce gets data from the cluster, performs operations, and publishes the results back to HDFS at the end of each phase. Due to the latency of disk I/O, MapReduce jobs are slower since each step involves a disk read and write.

Resilient Distributed Dataset (RDD), which is a cached collection of objects that can be reused across numerous Spark computations. This reduces latency substantially, making Spark several times faster than MapReduce, especially when performing machine learning and interactive analytics. Apache Spark is a distributed processing solution for big data workloads that is open-source.

3.3 Openclean

Openclean is a Python data cleaning and profiling module. The initiative was inspired by the fact that data preparation remains a significant barrier for many data science undertakings. Profiling is required to understand data quality issues, and data manipulation is required to change the data into a format that is suitable for the intended purpose.

While many different tools and strategies for profiling and cleaning data have been developed in the past, one major difficulty we notice with these tools is the lack of access to them in a single (unified) framework. Existing tools may be written in a variety of programming languages and require a significant amount of time and effort to install and use. In some circumstances, promising data cleaning approaches have been reported in the scholarly literature, but no appropriate codebase exists. We believe that the absence of seamless access to prior work is a big factor in the long time it takes to prepare data.

Openclean's purpose is to bring together data cleaning technologies in a single environment that is simple and straightforward for data scientists to utilize. Users can use openclean to create and execute cleaning pipelines using a range of different tools. We want openclean to be flexible and extendable so that new features may be easily added. To that purpose, we define a set of primitives and APIs for the various types of openclean pipeline operators (actors).

3.4 Profiling

The process of evaluating, analyzing, and developing relevant summaries of data is known as data profiling. The procedure generates a high-level overview that aids in the identification of data quality concerns, risks, and overall trends. Data profiling provides key data insights that businesses can use to their advantage. In particular, sifts through information to identify its legitimacy and quality. Analytical algorithms analyse data in minute detail by detecting dataset properties such as mean, minimum, maximum, percentile, and frequency. The program then conducts analysis to reveal metadata such as frequency distributions, key relationships, foreign key candidates, and functional dependencies. Finally, it applies all of this data to show how those aspects correspond with your company's standards and objectives.

Data profiling can help you avoid costly mistakes in your client database. Null values (missing or unknown values), values that shouldn't be included, values with unusually high or low frequency, values that don't fit expected patterns, and values that are outside the normal range are all examples of these errors.

3.5 Data Cleaning

The practice of correcting or deleting incorrect, corrupted, improperly formatted, duplicate, or incomplete data from a dataset is known as data cleaning. There are numerous ways for data to be duplicated or mislabeled when merging multiple data sources. Even if the data is right, outcomes and algorithms are untrustworthy if the data is erroneous. Because the methods will differ from dataset to dataset, there is no one-size-fits-all approach to prescribing the exact phases in the data cleaning process. However, it's critical to create a template for your data cleaning procedure so you can be sure you're doing it correctly every time.

4 Similarity Measurement

A few steps were involved in the procedure of determining similarity in our method:

4.1 Creating a dataset abstract

Since the most naive approach was to load all the datasets from NYC Open-Data to compare similarity, we had to compare the column between our starting dataset and all the datasets from the overall. Unfortunately, due to the size of the overall datasets, it would take a while to load all the data columns. Thus, taking into account the scale of the data, we generated an abstract of the dataset before we measured the overlap. During the generation of this abstract, a spark program was used; each row of the abstract began with the filename, followed by all columns, separated by commas.

In our first step, we created a dataset name list containing all the dataset paths on the HPC server using the hdfs command. A pyspark program was used to load all the datasets one by one on the server using this file, which was an index of the dataset path. Different from loading all the datasets and comparing their similarity, in our similarity measurement design, we manually check the columns of each dataset and compare them with the columns that our original dataset has.

4.2 Overlapping datasets analysis

Dataset abstracts make it easy to measure the overlap of datasets locally. The starting dataset contains all the columns of "311 Service Requests from 2010 to Present". Then we started applying the logic that we have already applied to our main dataset and checked that if the anomalies from the abstract dataset has been removed or not.

4.3 Collection of datasets used

The most common column in the datasets with overlap was 'Borough', hence we examined them in the similarity measurement results and we also analyzed some other columns such as Incident Zip code, StreetName, etc, and selected 14 datasets that had multiple overlap columns with our starting dataset. We used the openclean library to examine the dataset names since we only knew the dataset filename. Below are the datasets:

qy7q-cb9e	Address Assignments
tdd6-3ysr	311 Call Center Inquiry
psqs-brte	Customer Service Module
i2y3-sx2e	Curb Metal Data
zmut-au2w	DYCD after-school programs: Immigrant Services
mu46-p9is	Forestry Service Requests
yvxd-uipr	Board of Standards and Appeals (BSA) Applications Status
Yv4m-nu6d	3K Projects by Site Locations

According to the similarity measurement, we picked four columns with high frequency among all the columns in the whole dataset collection, which were: Agency Name, Borough, Incident Zip, City.

5 Methods, architecture, and design

We have employed an open-source library named openclean for most of our data profiling and data cleansing. We also explain our approaches alternative to the openclean library standard techniques and why they may be effective. We faced some challenges while cleaning and thus they were mentioned and how we overcame them was also mentioned.

We performed data profiling to get a proper idea about the structure of our dataset and so we can further formulate our cleaning strategies. We used the openclean library function 'profile' to perform data profiling. We also drop duplicate rows from our dataset if any as they may only hinder our analysis.

We see that many of the data values of the attributes have 'Unknown' and 'Unspecified' values. So we replace such values with null values. We found out that attributes such as Intersection Street 1', 'Intersection Street 2', 'Landmark', 'Facility Type', 'Due Date', 'Park Facility Name', 'Vehicle Type', 'Taxi Company Borough', 'Taxi Pick Up Location', 'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp' and 'Bridge Highway Segment' had missing values greater than 50

We noticed that for our dataset there were duplicate columns such as 'Park Borough' and 'Borough' so we remove the "Park Borough' column as we noticed that the 'Borough' column is more common for several datasets We also noticed that Location Column can be derived from combination of longitude and latitude so we also removed Location as a part of Dimensionality Reduction.

5.1 Date Columns ('Created Date', 'Closed Date', 'Resolution Action Updated Date')

After the dataset has been profiled and some modifications were made, we now move to target each attribute and clean it.We provide an introduction for the attribute, the cleaning strategy, and the cleaning results and challenges faced if any.

5.1.1 Column Introduction

'Created Date' and 'Closed Date' are common for several datasets.
Created Date: Created Date is the date when the service request was created.
Closed Date : Closed Date is the date when the service request was closed by the responding agency.

5.1.2 Data Insights and Problems

We find that created date has no null or missing values. Close date and Resolution Date actually have same dates for corresponding rows but also contain missing values.

5.1.3 Cleaning Strategy

1. We remove all entries where the closed date, resolution action action date both are null and status is either closed or null.
2. We filled the missing closed date values based on the resolution action update date and vice versa.
3. We removed all the entries where closed date $<$ created date and created date $>$ current date.

5.1.4 Results

We found this strategy to work for our dataset because we had no nulls or different date format's for same column.

5.2 Agency and Agency Name

5.2.1 Column Introduction

Agency: It is the acronym form of the responding City Government Agency Agency Name: It is the full name of the responding City Government Agency.

5.2.2 Data Insights and Problems

No mising or null values for both coloumns. We found that ”MAYORâ x80 x99S OFFICE OF SPECIAL ENFORCEMENT” in agency should have been ”OSE” as short form and ”Mayor’s Special Office of Enforcement” as complete Agency Name. We also found a functional dependancy of one-to-many relationship between Agency and Agency Name.

5.2.3 Cleaning Strategy

1. We replaced the above mentioned discrepancies for Agency and Agency Name with the correct values.
2. We cross checked the functional dependancy and found it to hold true across our dataset, hence indicating the datset is clean.

5.2.4 Results

We found this strategy to work for our dataset because we had no nulls or missing values and the functional dependacy between columns also to hold true.

5.3 Complaint Type

5.3.1 Column Introduction

It is the type of complaint of the service request.

5.3.2 Data Insights and Problems

We found various types of complaints and on grouping them we found that some of the common complaints like Noise were again having been present in various other complaint names like Noise - Commercial etc., We also found that there very similar complaints with just few special character differences or single letter differences like Derelict Vehicle and Derelict Vehicles etc.,

5.3.3 Cleaning Strategy

1. To group complaint type with having difference of characters greater but are similar we used regex logic. This strategy helped us group all noise, traffic , Ferry , Taxi complaints under single complaint types.
2. To deal with similar sounding word we use KNN clustering algorithm with similarity function of LevenshteinDistance and prediction with greater than 0.75. We then replaced the values with cluster suggestion.

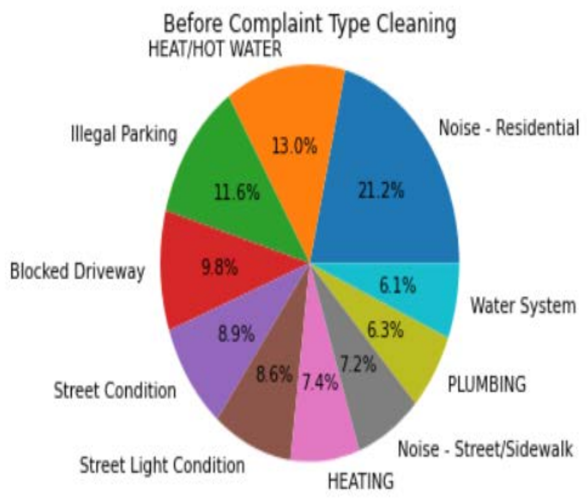
5.3.4 Result

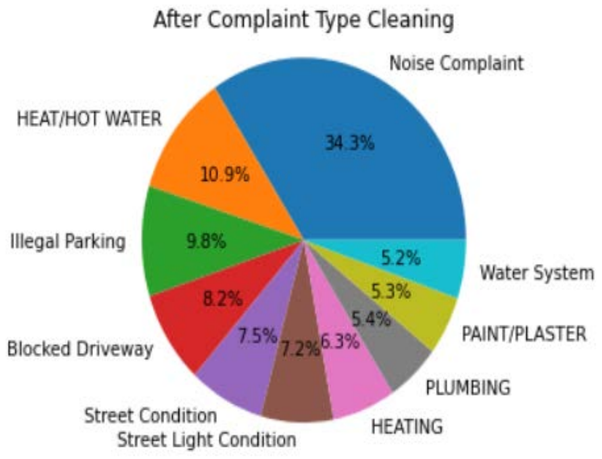
We were able to group many of important complaint types under single types. KNN cluster with similarity function of LevenshteinDistance was able to group some of the categories correctly but also failed at some cases. Hence the decreasing the precision and recall values.

5.4 Incident Zip

5.4.1 Column Introduction

It is the Incident location zip code.





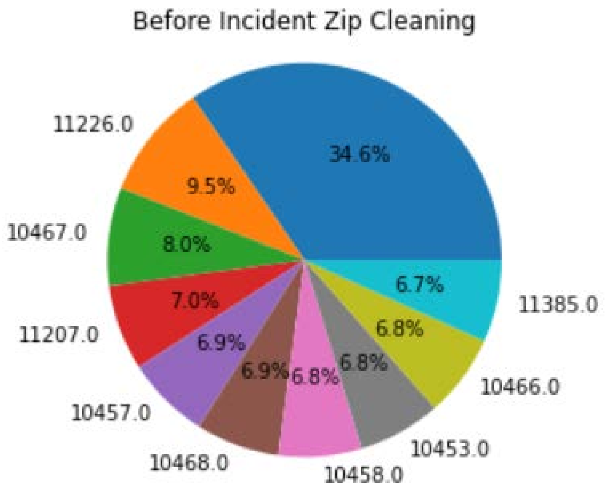
5.4.2 Data Insights and Problems

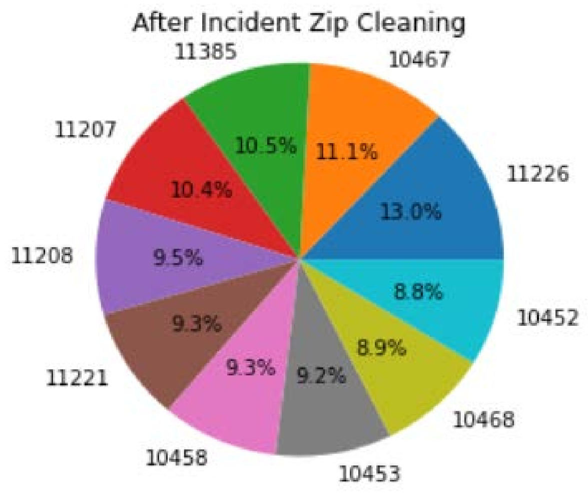
On inspection we found that incident zip had many mixed type of values ranging from alphanumeric, inconsistent lengths, zip codes not form New.

5.4.3 Cleaning Strategy

We wanted to have a base column which should be able to determine any other location based column like Longitude,Latitude,Borough,City. Since zip code is a integer value and minimal constraints we decided to have zip as our base column for further cleaning on other columns.

1. Zfill with length of 5
2. Deleting rows where zip code is not numeric





3. Filtering only zip code values from 10001-11697 (New York Zip codes)

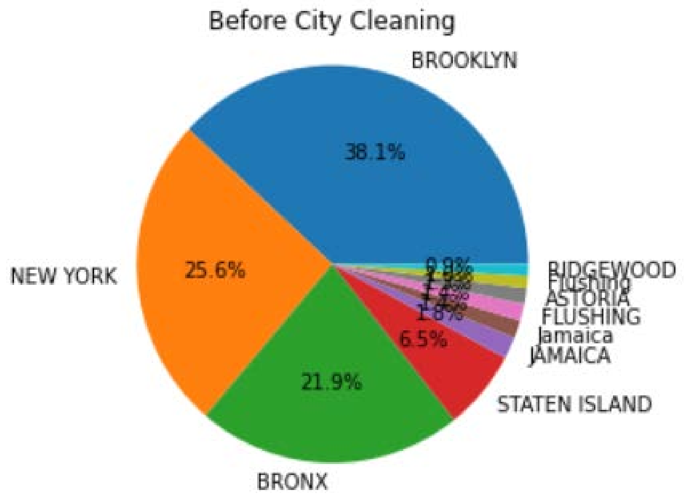
5.4.4 Results

Since we decided Zip code to be our base column for any further location based cleaning we find this strategy to work perfectly on our dataset.

5.5 City

5.5.1 Column Introduction

It is the City of the incident location.



5.5.2 Data Insights and Problems

Same values are represented in multiple cases like CORONA and corona. Minor spelling mistakes in city names are found. We also found that some zip codes have multiple values.

5.5.3 Cleaning Strategy

1. We first standardize the values of the city by stripping the values and capitalizing them.
2. As a part of initial strategy we moved forward with using soundex to correct the misspelled city names. But we found that some city names have same soundex like New york and Newark and since both these values were in huge numbers we could not have neglected any of these values.
3. As part of our next strategy we used functional dependency between the City and Zip code. We found out the most frequent city name for each zip code and replaced the other inconsistent values of the zip code with the correct value(i.e the value with the most frequency for the given zip).

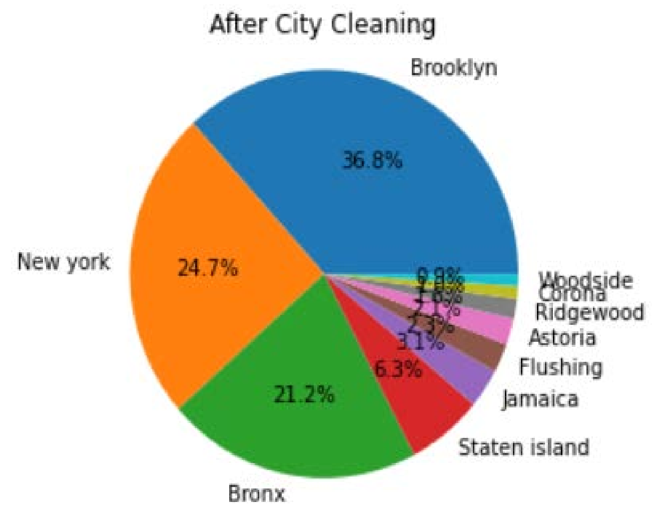
5.5.4 Results

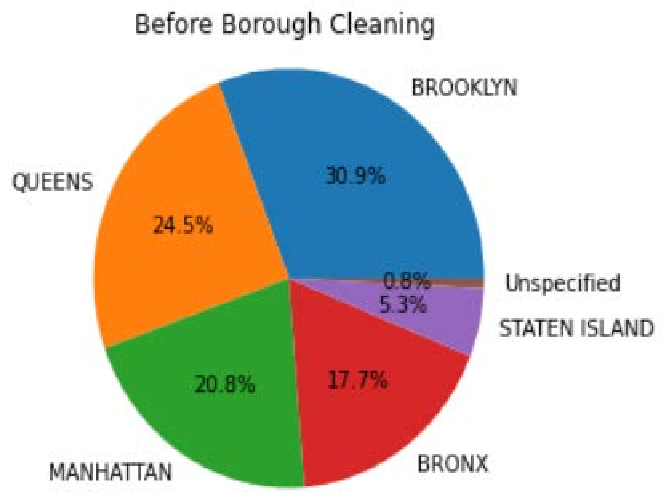
This strategy worked well since our data was huge and taking the assumption that correct values dominated the incorrect values in frequency.Probability of happening otherwise for huge data is unlikely ,in case that happens our strategy is not proficient.

5.6 Borough

5.6.1 Introduction

Contains Boroughs of New York City.





5.6.2 Data Insights and Problems

We found that data had numeric and then followed by Borough. There were also some value of 'UNSPECIFIED'. Some were having zip code -¿ Borough Violations as well.We find that there are Borough's apart from "Queens", "Bronx", "Brooklyn", "Manhattan", and "Staten Island".

5.6.3 Cleaning Strategy

1. Since we had a cleaned Zip code and also have functional dependancy between zip and city , we used a reference dataset which has mapping of zip to city names and used it replace UNSPECIFIED and alphanumeric values.
2. Then we filtered the values of Borough in list "Queens","Manhattan", "Brooklyn","Staten island","Bronx"

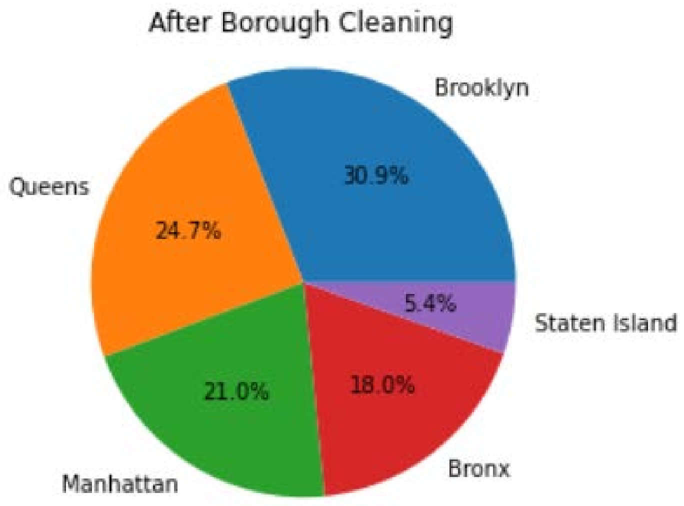
5.6.4 Result

This strategy has best precision and recall because we had direct mapping of 2 column and other column being completely cleaned before hand.

5.7 Incident Address, Street Name, Cross Street 1 and Cross Street 2

5.7.1 Column Introduction

Incident Address: It is the house number of incident address provided by submitter. Street Name: It is the street name of incident address provided by submitter. Cross Street 1: It is the first cross street based on the geo validated incident location. Cross Street 2: It is the second cross street based on the geo validated incident location.



5.7.2 Data Insights and Problems

We found that many of the values were similar with either the words jumbled or additional spaces between the words.

5.7.3 Cleaning Strategy

1. The best way to find out how to group such values with their correct representations would be to cluster them.
2. We cluster the values based on their similarity using open cleans Key Collision Clustering with default fingerprint key function.
3. We create a User Defined Function (UDF) to assign the highest value of a particular cluster to all the other values of the cluster.

4. There is also a possibility that the most frequent value may have many spaces. So we remove multiple spaces from the values if any.

5.7.4 Result

This strategy work values is most frequent otherwise we tend to replace correct with incorrect values but probability of happening this is very low and if it happens so can be changed manually later on in analysis.

5.8 Latitude and Longitude

5.8.1 Column Introduction

Latitude: It is the geo based Latitude of the incident location. Longitude: It is the geo based Longitude of the incident location.

5.8.2 Data Insights and Problems

Contains Nulls. As longitude and latitude are integer type we had an outlier analysis to plot any discrepancies and found none.Just to make sure the data is already clean by now we also ran an Functional dependacy violation algorithm using open clean between (latitude,longitude) -¿ Borough and found no violations.

5.8.3 Cleaning Strategy

Data was already in pure form without outliers, missing data or Violation of functional dependancies.

5.8.4 Result

With this column we end our cleaning process.

6 Conclusions

1. The dataset was profiled extensively and worked upon by using data cleaning techniques. Below are the objectives we have achieved while working on the dataset.
2. We generated reference data using our data cleaning techniques which can be employed further to validate data in other datasets.
3. We applied our strategies and scaled them to be adaptive and work on the other similar datasets with overlapping columns.
4. We have proposed unique methodologies in order to clean the columns and compared it to the simple approach.
5. We ran our techniques on sampled data and then scaled it to work on the whole dataset using Spark.
6. We have targeted each column to specifically find issues in each and work to clean them.

7 Our Contirbution

1. Based on all the analyses performed, the NYC311 Service represents a very popular and reliable channel and resource for the NYC communities to raise awareness to the local agencies and citizen services providers about multiple topics of importance and well-being for the society.
2. We were able to identify major issues in data and quality of the data. We have addressed these issues objectively.
3. We have formed a cleaned dataset in the csv format at the end of our cleaning and created a pull request to https://github.com/VIDA-NYU/reference-data-repository for further use.
4. All the code which include the spark code, jupyter notebook, visualization code and cleaned data is available on our GITHUB repository.
5. The reference data which we used for the Zip code is also included in our repository.

8 Challenges faced and Limitations

We have extensively implemented our methods on each attribute for appropriately cleaning the dataset. We faced some challenges while performing the overall cleaning of the dataset. The issues can be found below:

1. The reliability of the reference data can be questioned as it may not cover all the values in our dataset and may output incorrect values.
2. The clustering algorithm we implemented using openclean may not always output the right result and may need further processing as in our case.
3. The date structure of the datasets may vary and hence arises a need for better date formatting techniques.
4. The functional dependency which is demonstrated in our implementation may not always hold and hence, we may need another technique to overcome this problem.
5. There is a need to incorporate proper street names using the coordinates by using some Map API as it will be able to provide better accurate data.

9 Future Work

The dataset on which we have worked upon is quite an important one and could provide lot of insights for tackling the requests better. Here we suggest some future work

1. We can use a Google Map API in order to better map the street names with the help of Zip Codes.
2. We can try to automate the cleaning tasks by creating a framework or a library to incorporate custom cleaning functions.

References

[1] Müller, H., Castelo, S., Qazi, M., Freire, J. (2021).From Papers to Practice: The openclean Open-Source Data Cleaning Library. Proc. VLDB Endow., 14, 2763-2766.

[2] Svyatkovskiy, A., Imai, K., Kroeger, M., Shiraito, Y. (2016, December). Large-scale text processing pipeline with Apache Spark. In 2016 IEEE International Conference on Big Data (Big Data) (pp. 3928-3935). IEEE.