Profiling and Cleaning NYC Open Data

Erica Chou
New York University
Brooklyn, New York
emc689@nyu.edu

Varshitha Chennamsetti
New York University
Brooklyn, New York
vc2209@nyu.edu

Qin Ying Chen
New York University
Brooklyn, New York
qyc206@nyu.edu

ABSTRACT

The overall objective of this project is to practice and develop solutions for profiling and cleaning the datasets made available on the the NYC OpenData site [1]. This project consists of two parts. In part one, the 311 Service Requests From 2010 to Present dataset [2] was first profiled for any data quality problems and then cleaned to obtain a better quality dataset. In part two, ten datasets with similar fields to the original dataset from part one are found, and the data cleaning solutions used on the previous dataset are refined, scaled, and applied to these newly found datasets. The effectiveness of the original and refined approaches are evaluated and compared using precision and recall.

KEYWORDS
big data, datasets, data cleaning, data profiling, pyspark

1 INTRODUCTION

As data becomes increasingly abundant and available on sites like NYC OpenData, it has also become increasingly difficult to ensure the quality of the data. One of the challenges that data scientists have to deal with on a daily basis is evaluating and cleaning the data they work with. This is important because regardless of how effective a data analysis model and solution is created to be, feeding the model poor quality datasets will result in poor analysis and incorrect conclusions.

With large datasets consisting of millions of rows, it is impractical to manually evaluate and clean the data. In response to this problem, techniques and applications have been developed to profile and clean big data in a more efficient manner. Several techniques make use of open-source frameworks such as Hadoop and Spark to scale the solutions for large datasets. If applied effectively, Spark and Hadoop allows for distributed and parallel processing of large datasets, speeding up the process of profiling and cleaning large datasets. This means less wait time and effort for data scientists who want better quality datasets for more accurate analysis and conclusions.

To profile and clean the datasets for this project, the python version of Spark, pyspark (v3.2.0), is used. Spark is chosen over Hadoop because it is easier to use, it can run on Hadoop, and it allows for SQL operations, streaming, and complex analytics. This project consists of two main parts. In the first part, the 311 Service Requests From 2010 to Present dataset [2] is profiled for any data quality problems and solutions are determined to obtain a better quality dataset. This is completed on a sample of around five million rows of the 27.1 million rows from the dataset, and Google Colab is used to run pyspark solutions. For the second part, the solutions are refined, scaled and tested on the original dataset as well as ten similar datasets found on NYC OpenData. The precision and recall of the original and refined approaches are also calculated to compare their effectiveness. The solutions for this part are tested and run on NYU's Hadoop cluster called Peel, made available by NYU High Performance Computing (NYU HPC) services.

2 PROBLEM FORMULATION

To determine data quality issues that are present in the 311 Service Requests From 2010 to Present dataset [2], different profiling techniques are applied during the exploration of the dataset. This dataset contains 27.1 million rows and 41 columns, where each row is a 311 service request. To simplify the profiling process, a sample dataset consisting of around five million randomly sampled rows from the entire dataset is created and used. The problems that are profiled for and identified can be broken down into the following sections: uniformity, accuracy, inconsistency, completeness, and outlier.

Uniformity in a dataset refers to ensuring that the data of the same column use the same format and unit of measure. To identify if the columns are uniform, each column is individually selected and reviewed for any non-uniformity. After profiling, a total of five columns are found to have values with non-uniform casing (i.e. some values are fully upper-cased, some are fully lower-cased, and some only have the first letter of each word upper-cased).

Accuracy in a dataset refers to ensuring that the data is close to the true, expected values. To identify any inaccuracies in the columns, a reference dataset is found with information such as US zip code, state name, and city name, and this dataset is used to compare with the 311 service request dataset to identify any possible inaccuracies, such as misspellings. By comparing the 311 service request dataset with the reference dataset, several inaccuracies are found in the city column of the 311 service request dataset.

Inconsistency in a dataset refers to data that contradict each other or data that are in a column that they are not expected to be in. Just as for the profiling for accuracy problems, a reference dataset is searched for to be used to compare with the 311 service request dataset. However, no complete dataset with this information could be found on the internet. As a result, a reference dataset is created with agency names and information. This reference dataset is used to compare with the 311 service request dataset to identify inconsistencies in agency and agency name columns.

Completeness in a dataset refers to whether or not the columns are completed with all required data. To evaluate the completeness of each column, the data in each column is grouped and counted. Several null (i.e. missing) values are identified in the city and address type columns.

Outlier in a dataset refers to any data that is significantly different from all other data. To identify potential outliers, the minimum and maximum dates in columns with timestamp types are evaluated.

Outliers in the dates are found in the closed date, due date, and resolution action updated date columns.

The above was a brief overview of the types of problems that the 311 service request dataset was profiled for. For more details and demonstration of the profiling techniques and the problems that are uncovered, please refer to the profiling notebook: *click here to redirect to colab notebook*. After profiling the dataset for problems, the next step is to identify solutions and techniques to clean the dataset for a new version that has better data quality.

3 RELATED WORK

With big data on the rise and the quality of data becoming an important problem to solve, several tools have been created to allow data scientists to more easily and effectively profile and clean data. One such tool is OpenClean [15], an open-source python library developed by the Visualization and Data Analytics Research Center at NYU (NYU VIDA) to assist with data profiling and cleaning. The creators of this tool found that despite there being several tools and techniques available for profiling and cleaning data, data scientists are often spending a long time and a lot of effort during this process to find, install, and integrate these different tools and techniques together in their projects. To resolve this problem, OpenClean provides a single framework for data scientists to use that is easy to understand and flexible for different uses during the data profiling and cleaning stage. The installation is also simple. OpenClean can be installed easily from PyPI, and there are several example notebooks provided in OpenClean's GitHub repository that demonstrates how profiling and cleaning can be done using this tool.

In addition to improving the quality of the data, it is also desirable to increase the amount of data because the larger the dataset, the better the model trained with the dataset will perform. This process of increasing the amount of data in a dataset with similar data is known as data augmentation. Data scientists can find similar datasets manually by searching for datasets that contain similar columns and data types, and once these datasets are found, they can be joined into one dataset for profiling, cleaning, and use. However, finding useful similar datasets can be time consuming and difficult. To make the search for similar datasets easier, the Visualization and Data Analytics Research Center at NYU (NYU VIDA) developed a project called Auctus [14], a web search engine for datasets. On a high level, this web search engine allows users to use their existing dataset to search for other similar datasets and provides the option to merge the existing dataset with the found datasets to form a new dataset. There are several libraries and services available on Auctus that can be used to help data scientists improve and ensure their dataset quantity and quality before feeding it into their model for analysis and inference.

4 METHODS, ARCHITECTURE AND DESIGN

To clean the dataset, PySpark, a Python interface for Apache Spark, was used. There were also two Reference Datasets that were used for cleaning. One is a dataset with all the location information i.e all cities, zip codes, latitude and longitudes of them. The link: *click here to download the described reference dataset*. The other reference dataset was created by us for agency name and agency acronym. The link: *click here to download the described reference dataset*.

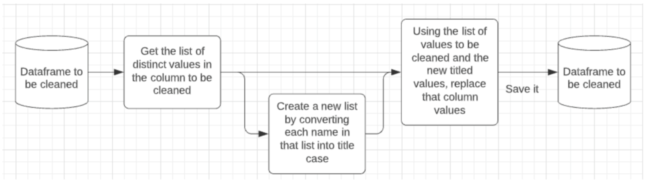
Dataframe to be cleaned
Get the list of distinct values in the column to be cleaned
Create a new list by converting each name in that list into title case
Using the list of values to be cleaned and the new titled values, replace that column values
Save it
Dataframe to be cleaned

Figure 1: System architecture for uniformity

Cleaning was sectioned off into five different categories: uniformity, accuracy, inconsistency, completeness, and outlier. Details about each section are shown below.

4.1 Uniformity

We describe uniform data as data that use the same format and unit of measure. To improve uniformity in the chosen datasets, columns that have string values are formatted into title case i.e, the first letter of each word will be capitalized and other letters are lowered. This is the first step that needs to be done before any other cleaning because profiling or cleaning other values will be varied otherwise. For example, both 'New York' and 'New york' are valid cities, but because there is no uniformity, one of them might be shown as a wrong or invalid city. This algorithm is further developed by getting rid of leading and trailing white spaces. These white spaces are removed using the trim function in PySpark.

As seen in Figure 1, the column to be cleaned is first converted to a list of distinct values. Then, a new list is created from that list by titling each string present in that list. These two lists are then used to replace the values in the column with the appropriate value from the new list and is then saved back to the dataframe.

4.2 Accuracy

We describe accurate data as data that is close to the true, expected values, and in terms of accuracy, there are many inaccurate pieces of data in the original dataset due to misspellings. For example, in the 'City' column, 'Brooklyn' is sometimes misspelled as 'Brookln' or 'The Brooklyn'. In these cases, when there is a column of values with the correctly spelled information, the correct column could be used to find similarities with the wrongly spelled values and replaced with the words that had the minimum distance like in the Figure 2. The distance between the strings was found through the Levenshtein distance. For the dataset with correct information, a reference dataset containing cities in New York was used. This can be applied to any column that has a dataset list of all correct values for that column.

4.3 Inconsistency

We describe inconsistent data as data that contains values that contradict each other or contains value that is not what we would expect based on the column that it is in. In terms of inconsistency, there might be cases where two columns might not match with the information they are conveying. For example, "Agency" and "Agency Name" columns should mention the same agency. In the

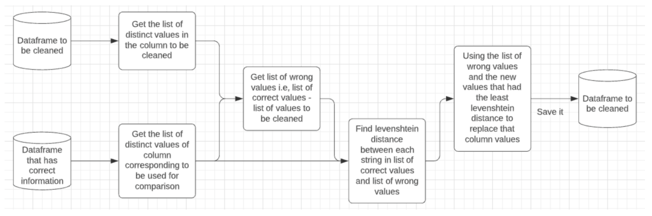
Dataframe to be cleaned
Get the list of distinct values in the column to be cleaned
Dataframe that has correct information
Get the list of distinct values of column corresponding to be used for comparison
Get list of wrong values i.e, list of correct values - list of values to be cleaned
Find levenshtein distance between each string in list of correct values and list of wrong values
Using the list of wrong values and the new values that had the least levenshtein distance to replace that column values
Save it
Dataframe to be cleaned

Figure 2: System architecture for cleaning misspelled data

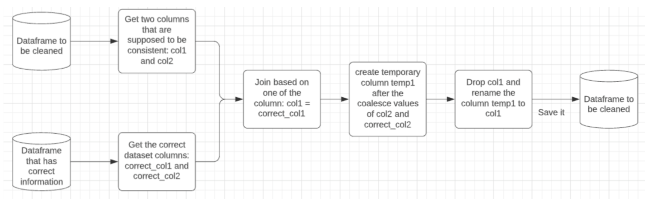
Dataframe to be cleaned
Get two columns that are supposed to be consistent: col1 and col2
Dataframe that has correct information
Get the correct dataset columns: correct_col1 and correct_col2
Join based on one of the column: col1 = correct_col1
create temporary column temp1 after the coalesce values of col2 and correct_col2
Drop col1 and rename the column temp1 to col1
Save it
Dataframe to be cleaned

Figure 3: System architecture for inconsistent data

chosen datasets with these two columns, “Agency Name” will be fixed by going through if there is a corresponding “Agency” in the same row that is also in the “correct” dataset. If there is a match, then we will change the value of the agency name to what it is supposed to be. This can be done with all columns that are supposed to be consistent with each other.

The way the cleaning is done for these inconsistencies is shown in Figure 3. It is assumed that there is a “correct” column between the two columns that are being checked for inconsistencies, in this example, it will be the column that holds the agency acronyms. The two columns are first joined on the basis of one of the columns considered and combining the other two columns with the “coalesce” function in PySpark. This function checks two columns, and if there is a value in both columns, the first column value is used. If only the second column value is available, then the new temporary column will attain the second column value. Finally, the actual column that was compared is dropped and this temporary value is renamed to the actual column name to be saved back to the dataset that is to be cleaned. This function can be further improved upon by checking first whether the column that is being used for the initial join does in fact hold correct values.

4.4 Completeness

We describe a complete dataset as a dataset that contains all required data. In terms of completeness, there are several null values in many of the columns for our datasets. If there is no other way to recover them, even with the dependent columns, then these null values are replaced with a placeholder within their own category in order. In this case, if the “Borough” column values can’t be found with any other dependent columns like “City”, then they are replaced with “Unspecified”, as this at least describes why the information is missing.

4.5 Outliers

In terms of outliers, there are many dates in the various different date columns of our dataset that either fall out of the date range stated by the dataset or are not believable dates that the events stated by the dataset could occur on. In such cases, those outliers are removed using a condition with what minimum and maximum values it is supposed to be because there is usually no available data that would help correct the date. To alleviate these problems, the ‘min’ and ‘max’ functions that are a part of the PySpark library are used for this purpose.

4.6 Multiprocessing

In order to run all 10 datasets at a time, the ‘multiprocessing’ package can be used. In this case, each dataset is quite different from each other and while some columns require cleaning, others don’t. So, each cleaning process is separately written in each python program file, and a new python program file will execute all files together by ‘pooling’ through the ‘multiprocessing’ and ‘os’ packages.

5 RESULTS

The following sections discuss the discoveries when applying the profiling techniques to each of the ten datasets that are found to be similar (i.e. have overlapping columns) to the 311 service request dataset, along with a brief description of the cleaning technique that is modified and applied to clean the dataset.

5.1 Profiling and Cleaning Similar Datasets

The Water Consumption And Cost (2013 - 2020) dataset [8] is similar to the 311 service request dataset in that both have a defined date range. For the Water Consumption And Cost dataset, the accepted date range is from "2013-01-01" to "2020-12-31". The profiling of this dataset reveals 732 rows with service start dates and 610 rows with service end dates that are outside this range. The service end date outliers are disregarded because it can be ended at any time. Cleaning is done to remove outliers in service start date such that a search for the minimum and maximum dates in the dataset show dates within the years 2013 and 2020. Another similarity between the water consumption dataset and the 311 service request dataset is the column with boroughs. Profiling the borough column of the water consumption dataset reveals rows with values that are not boroughs, corresponding to an accuracy problem. During cleaning, these rows, i.e. “FHA” and “NON DEVELOPMENT FACILITY”, are combined and changed to “UNSPECIFIED”.

The Heating Gas Consumption And Cost (2010 - 2020) dataset [7] is similar to the 311 service request dataset in that it also has a defined date range. For the heating gas consumption dataset, the accepted date range is from "2010-01-01" to "2020-12-31". The profiling of this dataset reveals 2731 rows with service start dates and 2603 rows with service end dates that are outside this range. Again, the service end date outliers are disregarded because the service can be ended at any time. Cleaning is done to remove outliers in service start date such that a search for the minimum and maximum dates in the dataset show dates within the years 2010 and 2020. The heating gas consumption dataset also contains a column for borough. Profiling this borough column reveals the same accuracy problem as described for the water consumption dataset. During cleaning, these rows, i.e. “FHA” and “NON DEVELOPMENT FACILITY”, are combined and changed to “Unspecified”.

The DOB permit issues dataset [4] is similar to the 311 service request dataset in that it also contains a column for cities. Profiling this dataset for accuracy in this cities column reveals that there are 12,527 rows with misspellings in the city names. After applying our uniformity solution to make the cities column uniform, the profiling for the accuracy in this cities column drops to 10,287. To clean this dataset and lower the number of misspellings in city names, the cleaning solution for fixing misspellings is applied.

The OpenRecords FOIL Requests dataset [12] is similar to the 311 service request dataset in that it has a column for agency name, and it also has a defined date range. Profiling the columns with the dates show that the created and submitted dates are within the range, and it is alright for the due date to be beyond this range. The agency name column is also found to be uniform, consistent, and accurate during the profiling process. Therefore, there is no cleaning that needs to be done on these columns for this dataset.

The Wholesale Markets dataset [9] is similar to the 311 service request dataset in that it contains a column for city names. Profiling this dataset for accuracy in this city column reveals that there are 46 rows with misspellings in the city names. After applying our uniformity solution to make the city column uniform, the profiling for the accuracy in this cities column drops to 13. To clean this dataset and lower the number of misspellings in city names, the cleaning solution for fixing misspellings is applied. Another column in the wholesale markets dataset that overlaps with the 311 service request dataset is borough (or the column labeled as BORO in the wholesale markets dataset). Profiling the BORO column reveals None values, indicating incompleteness. Just as with all other datasets that have this problem, we use our cleaning solution that updates the None values to "Unspecified" because we do not want null values but we also do not want to remove data.

The Bedbug Reporting dataset [11] is similar to the 311 service request dataset in that it contains columns with date values and a column with boroughs. Profiling the date columns show that the dates are all within a reasonable range, so there are no observable outliers. Profiling the borough column shows that they are all accurate and consistent. Therefore, there is no need to clean these columns for this dataset.

The Appeals Closed In 2018 dataset [10] is similar to the 311 service request dataset in that it contains date columns and a defined date range. Since this dataset suggests that only the closed dates are in 2018, the filed dates are not checked because there is no requirement on that to be within a specific year. Profiling the closed date column for minimum and maximum dates show that all values in this column are within the year 2018. Further profiling that checks if there are any rows where the dates filed is later than the date closed reveal that there are no such rows, so all the rows have valid dates. However, the profiling that checks to see if there are any rows that have status flag as "Open" and has a date closed value show that there are two rows that satisfy this condition, resulting in an inconsistency problem in this dataset. To fix this problem, we apply our cleaning solution where we check the status column, and if the status column is inconsistent with the date closed column, we update the status column to make them consistent.

The Active-Projects-Under-Construction dataset [3] is similar to the 311 service request dataset in that it has both a borough and city column. Profiling the borough column revealed uniformity problems where there are spaces in front or back of the names that need to be removed. The number of distinct rows for this borough column is found to be 14 before the cleaning is applied. This number drops to 11 once the uniformity cleaning solution is applied to this column. In addition to the uniformity problem, there are also accuracy problems in the borough column where sometimes a "K" is written in place of "Brooklyn" and in the city column where it contains borough values even though it is for city. Since the accuracy problem in the borough column is specific to Brooklyn, we use the cleaning solution that directly fixes the problem by replacing "K" to "Brooklyn" in all relevant rows. As for the city column that contains borough values, we could apply a cleaning solution to change all the boroughs to "New York", but since we have also seen city values as boroughs before, we have decided to keep this column as it is.

The Greenbook dataset [6] is similar to the 311 service request dataset in that it has a column for agency acronym and agency name. Profiling the agency acronym column uncovered a completeness problem where there are 1,797 null values. Profiling the agency name and agency acronym columns using the custom reference dataset uncovered incompleteness and inaccuracies in agency name. Since the reference dataset for agency and agency acronyms are not fully complete, it can be observed that valid agency acronyms are marked as incorrect. The profiling results that reflect this are ignored. As for the agency name results, it can be observed that there are clear inaccuracies. For cleaning, we attempt to modify and apply our solution such that we can use the agency acronym column to update and correct agency names. However, this means that we are unable to fix the ones that have null value in the agency acronyms column, and thus this solution is not recommended.

The DOHMH New York City Restaurant Inspection Result dataset [5] is similar to the 311 service request dataset in that it has columns with date values and a column with the boroughs, called BORO. Profiling the columns with the date values show that the dates are reasonable. It should be noted that for the inspection date column, there are dates in the year 1900 which seem like outliers; however, reading the description reveals that these dates indicate new establishments that have not been inspected yet, so these are not outliers. Profiling the BORO column reveals an inconsistency problem where the majority of the borough values are strings but there are also rows with 0 as the borough. To fix this problem, we apply the cleaning solution where we replace the 0 with "Unspecified" to match the expected type.

For better visualization of the inaccurate data that we encountered during profiling and the effect of applying uniformity cleaning, refer to Figure 4, 5 and 6.

5.2 Precision and Recall

For comparing the results of the functions written, precision and recall is used. Precision gets the intersection of what documents are supposed to be shown and what was shown by the function implemented over all the documents the function is showing. Recall is the intersection of what documents are supposed to be shown and what was actually shown by the function implemented over all the documents that are supposed to be shown. For example, while profiling the values in the column "Complaint Type", regular

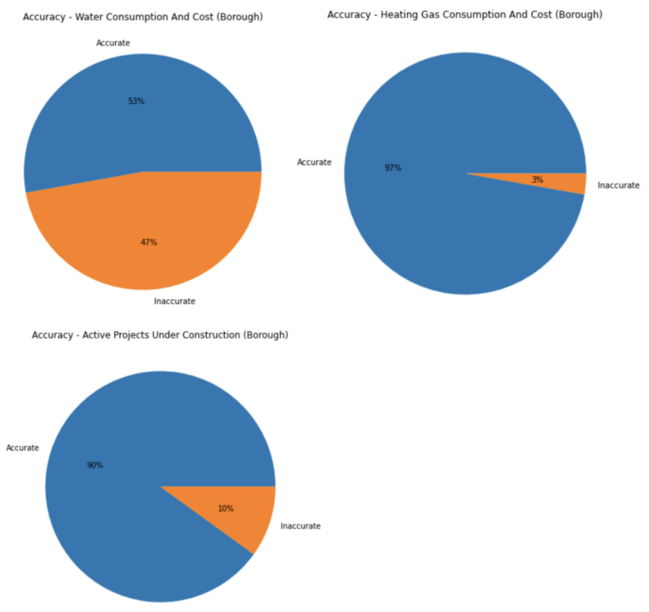

Figure 4: Pie charts showing the profiling done on different data frames that contain the column 'Borough'. From left to right and top to bottom, the charts correspond to Water Consumption And Cost, Heating Gas Consumption And Cost, and Active Projects Under Construction respectively. The blue coloured parts represent the percentage of accurate data while the orange parts represent the inaccurate percentage.

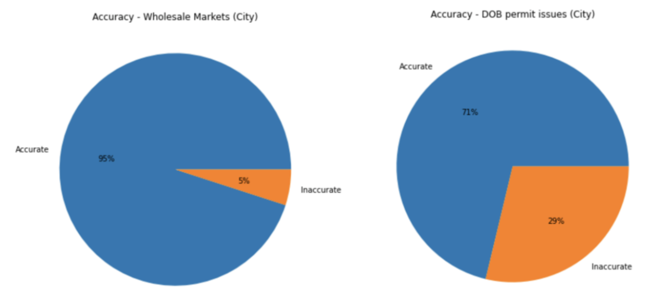

Figure 5: Pie charts showing the profiling done on different data frames that contain columns 'City'. The orange portion of the pie chart represents the inaccurate data count whereas blue coloured portions represent the total count of rows in that column that have values.

expressions are used to see which values are not in the correct format. However, sometimes there are values that are written in correct format but are recognized as in the wrong format because they matched the regular expression. The equations for Precision (1) and Recall (2) are shown below:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{1}$$

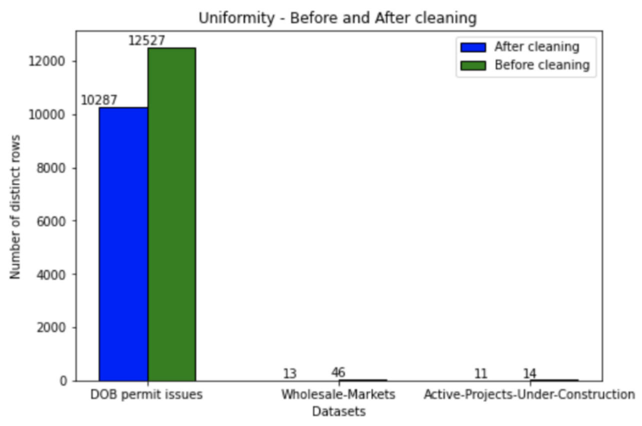

Figure 6: Bar graph showing the count of distinct rows of columns before and after cleaning. Blue coloured plots represent the distinct rows count after cleaning whereas green coloured bars represent the distinct rows count before cleaning. The column that was cleaned were city columns in both 'DOB permit issues' and 'Wholesale markets dataset'. The Borough column was cleaned in the 'Active projects under construction' dataset.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2}$$

Confidence interval is used to determine the sample size for calculating precision and recall. The subsampling size was found through confidence interval and confidence level. The confidence level determines how many sample rows to take in order to get that much confidence. For example, if there are 10,885 distinct rows in the column to be cleaned with a confidence interval of 5, it would be 371 rows. A sample size calculator can be used to determine the required sample size [13].

The precision and recall for the DOB permit issues dataset was found by using a subsample and the subsampling size was calculated to be 371. The precision and recall was used for cleaning the misspellings. The way the algorithm classifies is by comparing the cities with a reference dataset full of city values, therefore, there is no scope for any false positives if the reference dataset is regarded as the ground truth. But there are times the cities may not be recognized as true positives because of the way it is being classified. For example, in the reference dataset, 'New york' is present as a city but 'New york city' is classified as a misspelling. In this case, it is a false negative. The precision and recall was also calculated on the distinct city values and checked if there are misspellings.

As mentioned above, precision will be 1 for both datasets as false positive is none. Recall for the column 'Owner's House City' in the DOB permit issues dataset is 0.92. The recall for the column 'City' in the Wholesale Market dataset is 0.87.

6 LIMITATION AND FUTURE WORK

Some future work that could be done on our algorithms is continuing to make our algorithms more generalized. Many of our cleaning algorithms that we created were very specific to cleaning the original dataset, and as a result, were difficult to try and generalize for other dataset use. For example, the method for verifying if an Agency Acronym matched the corresponding Agency in a row also required a unique key field inorder for the join between the reference data and dataset that needs to be corrected to work. However, datasets such as Greenbook did not have this field, which made trying to make this method fit this dataset impossible.

There were cases where the 'city' column values had 'Borough' names which might result in inaccurate cleaning since the cleaning is done using a Reference dataset that only contains valid city names. This is because borough names such as Manhattan, Brooklyn, Staten Island, Queens, and Bronx are all a part of New York City and do not count as cities. There were also cases where 'New york city' was cleaned as 'New York Mills' because the levenshtein distance is used for cleaning the column which will see lesser distance when comparing 'New york mills' with 'New york city' than when compared with 'New york'. This algorithm could be improved so that there are lesser errors in cleaning the data.

Another limitation that was faced was the environment set up. Due to the large amount of data and the limited resources available, there were often java heap space issues when running the code. The solution that is currently in use every time a java heap space issue is faced is to either increase the spark executor memory or restart the session completely. However, these are not practical/long-lasting solutions. One possible way to fix this problem could be to further optimize the algorithm to deal with the large amount of data or parallelizing the algorithms to reduce the strain put on the system. By further breaking down and optimizing the use of the java heap space, less time could be spent on waiting for the code to run and then eventually crash.

REFERENCES

[1] NYC Open Data. [n.d.]. *NYC Open Data.* City of New York. Retrieved December 12, 2021 from https://opendata.cityofnewyork.us/

[2] NYC Open Data. 2011. *311 Service Requests from 2010 to Present.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

[3] NYC Open Data. 2011. *Active Projects Under Construction.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Housing-Development/Active-Projects-Under-Construction/8586-3zfm

[4] NYC Open Data. 2013. *DOB Permit Issuance.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Housing-Development/DOB-Permit-Issuance/ipu4-2q9a

[5] NYC Open Data. 2014. *DOHMH New York City Restaurant Inspection Results.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j

[6] NYC Open Data. 2015. *Greenbook.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/City-Government/Greenbook/mdcw-n682

[7] NYC Open Data. 2016. *Heating Gas Consumption And Cost (2010 - 2020).* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Housing-Development/Heating-Gas-Consumption-And-Cost-2010-2020-/it56-eyq4

[8] NYC Open Data. 2016. *Water Consumption And Cost (2013 - 2020).* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Housing-Development/Water-Consumption-And-Cost-2013-2020-/66be-66yr

[9] NYC Open Data. 2017. *Wholesale Markets.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/City-Government/Wholesale-Markets/87fx-28ei

[10] NYC Open Data. 2019. *Appeals Closed In 2018.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/City-Government/Appeals-Closed-In-2018/uetw-jfrg

[11] NYC Open Data. 2020. *Bedbug Reporting.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/Housing-Development/Bedbug-Reporting/wz6d-d3jb

[12] NYC Open Data. 2021. *OpenRecords FOIL Requests.* City of New York. Retrieved December 12, 2021 from https://data.cityofnewyork.us/City-Government/OpenRecords-FOIL-Requests/kegn-anvq

[13] Creative Research Systems. 2012. *Sample Size Calculator.* Creative Research Systems. Retrieved December 12, 2021 from https://www.surveysystem.com/sscalc.htm

[14] VIDA-NYU. [n.d.]. *NYC Open Data.* New York University. Retrieved December 12, 2021 from Auctus

[15] VIDA-NYU. [n.d.]. *openclean - Data Cleaning for Python.* New York University. Retrieved December 12, 2021 from https://github.com/VIDA-NYU/openclean