CS-GY 6513 Big Data Fall 2021 Project - Group 6

Minghao Shao
ms12416@nyu.edu
New York University
New York, NY, USA

Haozhong Zheng
hz2675@nyu.edu
New York University
New York, NY, USA

Hanqing Zhang
hz2758@nyu.edu
New York University
New York, NY, USA

KEYWORDS

data sets, New York City, data profiling, data cleaning, effectiveness, reference data

ACM Reference Format:
Minghao Shao, Haozhong Zheng, and Hanqing Zhang. 2021. CS-GY 6513 Big Data Fall 2021 Project - Group 6. In *Proceedings of Big Data (CS-GY 6513).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In our daily life, we may encounter a huge number of datasets with different quality. In some datasets, values of certain attributes were missing while in some datasets, some values were incorrect in people's cognition. In which case data profiling and data cleaning were necessary, which formed two main approaches in our project. In this project, we would use data profiling techniques, a process of validation on the data from available sources [1], to analyze the data structure, check the effectiveness of each data and inspect the defect of the data set hunted and mined by us. And we would apply our strategies to clean the defected data, starting from a specific assigned data set and extending them to multiple data sets we discovered.

We took the dataset "Citywide Payroll Data (Fiscal Year)" as our starting point, and mined the data sets provided by NYC OpenData. Profiling and cleaning the potential low quality data and analyzing, visualizing them, and generating useful potential reference data.

2 PROBLEM FORMULATION AND OBJECTS

From our data mining procedure, we may solve the problems below:

- Starting from the dataset "Citywide Payroll Data (Fiscal Year)", how can we find the datasets with overlap fields via data hunting?
- How can we apply our data clean strategy to the datasets we found?
- How well does our original data clean strategy work? What more can we learn from these datasets or what's the difference between our original dataset and new datasets?
- How can we improve our strategy to increase the effectiveness of data clean?
- How to generate reference data which could be used for future data clean tasks?
- How to extend our strategies to all the datasets we may find?

3 PRIORI KNOWLEDGE AND RELATED WORK

In this section, all the prior knowledge, and all the related work done by other authors used by us were introduced.

3.1 Prior Knowledge

3.1.1 City Agencies in NYC. City agencies in NYC were composed of dozens of city departments in New York City. Each agency in our project datasets has its unique formal agency name, each of these agencies has its own department with a unique abbreviation, which means different agencies may belong to the same department under the New York City government. The reference link was a list of NYC agencies [2]. Reference data: Reference data was a kind of data used to validate or classify other data [3], such as country code, the borough name, or the abbreviation of the New York City department name.

3.1.2 Effectiveness, precision and recall. In this project, effectiveness was used to measure how well our original data clean strategies work, including precision and recall. In our definition, precision was the fraction of the real problem data that should be cleaned among all the data we cleaned, while recall was the fraction of the data detected and cleaned by our strategies among all the problems spotted by us.

The formula for calculating the precision shown below:

$$precision = \frac{|(Problemsindata \cap Problemsourstrategiesfound)|}{|Problemsourstrategiesfound|}$$

The formula for calculating the recall shown below:

$$recall = \frac{|(Problemsindata \cap Problemsourstrategiesfound)|}{|Problemsindata|}$$

3.2 Related Work

3.2.1 NYC OpenData. All the datasets used in this project were available on the NYC OpenData platform and obtained from NYU HPC cluster. NYC OpenData is a free platform involving public data published by New York City agencies and other partners [4], containing datasets from different sources collected and used by the city government. The data provided by this platform covers vast fields such as traffic violations, payroll information and public services data.

3.2.2 Spark and Openclean. Two of the main platforms this project is based on. In this project, all the data analyzing tasks involving large scale data were performed on NYU HPC by Apache Spark platform, which was an open-source unified analytics engine for large-scale data processing [5]. Openclean was also used as our data profiling and clean library developed by NYU VIDA, which contains necessary data profiling and cleaning tools such as Pandas and KNN cluster [6], which was widely used in this project.

3.2.3 Profiling and clean tasks on starting data set [7]. Since our original strategies have been applied on the dataset "Citywide Payroll Data (Fiscal Year)", in this project, we would extend our strategies to the new datasets we found, comparing their difference and improve our data clean strategies to make it generalize to more datasets. Our previous profiling and clean could be found here.

4 DATA SETS

We started from the data set "Citywide Payroll Data (Fiscal Year)" which was a data set containing the City's budget used on employees' salary and overtime pay [8] and was used to analyse the allocation of city's financial resources Below was the column information of this dataset:
Agency, Last Name, First Name, Middle Initial, Agency Start Date, Work Location Borough, Job Title Description, Leave Status as of the close of the FY (June 30th), Base Salary, Pay Basis, Regular Hours Paid, Regular Gross Paid, Overtime Hours worked, Total Overtime Paid
From the data columns above, it could be profiled that this document covers a variety of fields which may overlap with other dataset regarding financial conditions of the city, including Agency, Borough, Date and Salary.
As our starting point, we inspected all the columns in this data set, and extracted these columns to compare with all the datasets provided by the NYC OpenData by measuring their column similarity which was described in the following section.

5 SIMILARITY MEASUREMENT

The procedure of similarity measurement in our method was divided into several parts:

5.1 Generate dataset abstract

For measuring similarity, it was necessary to compare the column between our starting dataset and all the datasets from NYC OpenData, hence the most naive approach was to load all the datasets from the overall. However, due to the size of the overall datasets, it would take a while to load the data columns. Hence in consideration of the scale of the data, before we measure the overlap, we generated a dataset abstract. The abstract was generated by a spark program, for each row of the abstract, the filename was recorded at the beginning followed by all the columns separated by comma symbols.
First we created a dataset name list using hdfs command, containing all the datasets path on the HPC server. This file was an index of the dataset path which was used by us in a pyspark program to load all the datasets one by one on the server. Different from loading all the dataset and comparing their similarity, in our similarity measurement design, only the columns in the dataset were loaded, by which the procedure of loading the whole dataset was avoided since for computing the overlap of the dataset fields, only the columns information was loaded. Next, as described above, the column information was stored in the dataset abstract with a proper size.

5.2 Analysing dataset overlap

With the dataset abstract, the overlap of the dataset could be measured locally. We extracted all the columns from our starting dataset, which was "Citywide Payroll Data (Fiscal Year)". Then the k-shingle method with Jaccard distance was used to measure the similarity between the starting dataset and all the datasets in the abstract.
There were two parameters in our similar measurement program, a k value, which was the number of shingles we would use to measure the similarity, and a t value, which was a threshold; only the columns with similarity larger than this threshold were recognized as 'overlap column'. With all the column string divided into shingles, jaccard distance was used to determine the final similarity. Since we only have to care about the columns which overlap with the starting dataset, the overall similarity between these datasets was not considered. In our experiment, 3 shingles were used and the threshold was set to 0.5. During the procedure of similarity measurement, all the columns or shingles were transformed into uppercase to avoid the loss caused by the letter case.
All the datasets name and the overlap column were recorded into a text file, which was used for us to determine the dataset we decided to handle.

5.3 Datasets collection used

With our similarity measurement, over 900 of datasets were found that had overlap columns with our starting dataset. Since we recorded the overlap columns, the overlap datasets could be inspected manually.
We found that most of the datasets with overlap had the column 'Borough', hence we inspected them in the similarity measurement result, and picked 14 datasets which had multiple overlap columns with our starting dataset. Since only the dataset filename was known by us, we used openclean library to inspect their name. These datasets were recorded below:

Dataset identifier	Dataset name
bty7-2jhb	Historical DOB Permit Issuance
ptev-4hud	License Applications
hy4q-igkk	311 Service Requests for 2006
aiww-p3af	311 Service Requests for 2007
3rfa-3xsf	311 Service Requests for 2009
m6ad-jy3s	FY18 BID Trends Report Data
emuv-tx7t	FY17 BID Trends Report Data
gt6r-wh7c	FY19 BID Trends Report Data
8eq5-dtjb	FY20 BID Trends Report Data
xrwg-eczf	SCOUT CORE
un8d-rbed	SBS ICAP Contract Opportunities - Historical
acdt-2zt9	2021 Open Data Plan: Future Releases
cwy2-px8b	Local Law 8 of 2020 – Complaints of Illegal Parking of Vehicles Operated on Behalf of the City
wye7-nyek	Interagency Coordination and Construction Permits Data (MOSYS)

According to the similarity measurement, we picked four columns with high frequent among all the columns in the whole dataset collection, which was: Agency Name, Borough, Date and Salary.

6 METHODS, ARCHITECTURE DESIGN AND RESULTS

In this section, we would introduce our strategies, applying our strategies to these overlap column we picked analyzing the result and refining our work.
All of the subsection under this section contains our profiling and cleaning procedure and refining of our work. Visualisation results were also included. We also noticed that there were some challenges and limitation on our data clean tasks, hence these reflections were pointed out and evaluated.

6.1 Agency Name

6.1.1 Column Introduction. Another column we cleaned is "Agency Name", the column of several datasets from NYC Open Data. In multiple different datasets, we can clearly find overlaps with the dataset we worked on. In this project, we examined these data sets and cleaned them with different strategies and tried to refine our techniques afterwards.

6.1.2 Our Original data clean strategy. Before applying the original strategy to all the selected datasets, we wanted to calculate the effectiveness of the old strategy, so we extracted the 'Agency Name' column and saved it as a new data frame. The original strategy is:

- We noticed that the "agency name" has different cases. Therefore, our strategy was to integrate them in capital letters.
- We noticed that there are many missing values in this column. Therefore, we entered the assigned values such as "OTHER" and "UNSPECIFIED" for these missing values, depending on the definition of "agency name" in the dataset.
- We used a KNN cluster to check for dataset input errors and fix them according to the results of the cluster.

6.1.3 Result of effectiveness and problems found. After applying the original strategy to the entire dataset collection, we inspected the cleaned-up dataset and checked the effectiveness of the original strategy, including the Agency Name column, which fetched almost none rows that were cleaned up. It turns out that the data in those datasets are already clean since there were no missing values or typo issues before we applied the original data clean strategy. After analyzing the effectiveness and profiling the cleaned-up columns with the original strategy, some issues were still identified. Some "Agency Name" values were shown in abbreviated form or abbreviated and full name mixed form, thus those data could not be determined by the original strategy, for example, we found 'NYPD' and 'NEW YORK CITY POLICE DEPARTMENT' separately, 'DEPARTMENT OF BUILDINGS (DOB)' and 'DEPARTMENT OF BUILDINGS' separately.

6.1.4 Our refined strategy. Depending on the remaining issue, we have improved our strategy to address these issues. Based on the problems we found, we refined our strategies to fix these issues:

- We created a reference data containing all the potential abbreviations and the 'Agency Name's formal name
- We used map to fix the abbreviated form or the mixed form problem based on the reference data we created

For generating add-map reference data, we extracted all the datasets we handled with both column 'Agency' and 'Agency Name'. We dropped all the values with the abbreviation problem which meant that their 'Agency' and 'Agency Name' were both in abbreviation form. We inspected this new mapping DataFrame, checked the missing values, and generated a copy of our reference data which would be used to extend our refined approach to other datasets.

6.1.5 Apply refined strategy and visualization. After improving the strategy, we found that the new strategy could identify the problem and clean up the problem on all the datasets we worked on in this project. Later, this new strategy was extended to apply to all 6 data records, including the Agency Name column. And then, we wanted to do a visualization of the cleaned data, but the result is not ideal or usable and we will explain the situation clearly in the next part.

6.1.6 Challenges and limitations. Though our new strategies could clean all the rows marked as dirty data by us, there were still some challenges and limitations for us:

- TAXI AND LIMOUSINE COMMISSION (TLC), NEW YORK CITY POLICE DEPARTMENT(NYPD), DEPARTMENT OF TRANSPORTATION (DOT), we can find that some part of them were represented by abbreviations and some of them are mixed form represented, though we created a reference data containing all the abbreviations we found and used map method, there might be more abbreviation form or mixed form used in other datasets which was not found by us, so there still can be some potential data with the same issue. Though we generated the reference data to mapping these abbreviations, it was still unknown whether there would be more to extend in this reference data.
- In some cases, the KNN cluster in openclean is not always reliable since it could not mark all the typo issues in values, and these typos only can be fixed manually sometimes. A more effective way could be used to make more precise predictions among all the data.
- We could not find a useable or effective way to visualize the "Agency Name" column because of the scale of the data and the gigantic amount of different Agency Name, for instance, we can find a lot of agencies belong to the same parent agency, but each appearance of the different sub agency are meaningful and we can't just merge them to make our visualized charts more readable, we have implemented a histogram visualization function in the notebook and we decided to discard the visualization plan for this column because of the futility.

6.2 Borough

6.2.1 Column Introduction. This column contains the borough in which the row of data was collected in New York City. In different datasets, the representation of this column could be different such as the case of the borough name and their abbreviation. In this project, we inspected these datasets and would clean them into uniformed format according to our strategy.

6.2.2 Our Original data clean strategy. Before we apply our original strategy to all of the datasets we picked, we extracted the borough column and saved it as a new data frame since we decided to calculate the effectiveness of our old strategy. Our original strategy was:

- Since we found that in our starting dataset, the case of the borough name was different, some were in letter case while some were in upper case. Hence our strategy was to uniform them into upper case.
- We found that there were a lot of missing values in this column, hence we filled these missing values with an assigned value such as 'OTHER' or 'UNSPECIFIED', depending on the definition of borough not sure in the dataset. After profiling the datasets, we filled the missing value with 'UNSPECIFIED'.
- We used KNN cluster to check typo mistakes in the dataset and fix them manually according to the cluster result.

6.2.3 Result of effectiveness and problems found. After applying our original strategy to the whole dataset collection, we inspected the cleaned dataset and among all 12 datasets containing column 'Borough', we got the effectiveness of our original strategy with precision 0.99967 and recall 0.99964 among all the 2086772 rows we cleaned and 2087274 with problem. We noticed that most of the problem was caused by the missing value and the case of the characters. Hence our original strategy produced a high effectiveness. After analysing the effectiveness and profiled the cleaned column with original strategy, some problems were still identified:

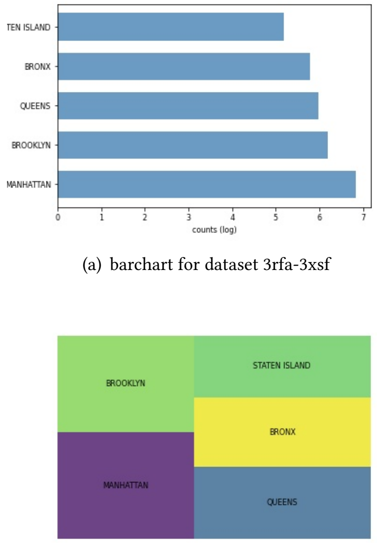
- Some borough values were represented in abbreviated form, hence our original strategies could not identify them.
- Though the KNN cluster detected some typo issues, such as 'Brooklyn' and 'Brooklyhn', there were still some values with type not detected.
- Some values were represented in zip code format, which could not be handled.
- Some values was a combine of several different or same brough names separated by semicolons or sharp symbols, which was confusing.

6.2.4 Our refined strategy. Based on the problems we found, we refined our strategies to fixed these issues:

- We created a reference data containing all the potential abbreviations and the borough's formal name.
- We adjusted different KNN hyper parameters, fixed these typo error manually after spotted out the typo error.
- We use another reference data which was able to map the all the zip codes and the borough this zip code belongs to [9].
- We split the value, took the first borough name as the value for all data with combined values.

6.2.5 Apply refined strategy and visualization. After refining our strategies, we noticed that our new strategies were able to detect all the problems and cleaned them among all the datasets we handled in this project.

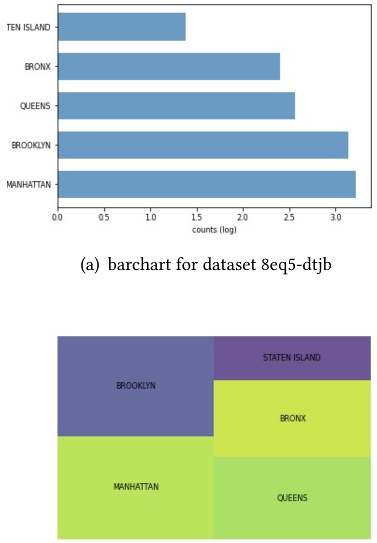
Then this new strategy was extended and applied to all the 12 datasets containing the column 'Borough'. The diagrams below were the visualization results of all datasets. In column 'Borough', log based barchart and treemap were used to represent their value counts and occupation in each of these datasets according to the borough.



(a) barchart for dataset 3rfa-3xsf

(b) treemap for dataset 3rfa-3xsf

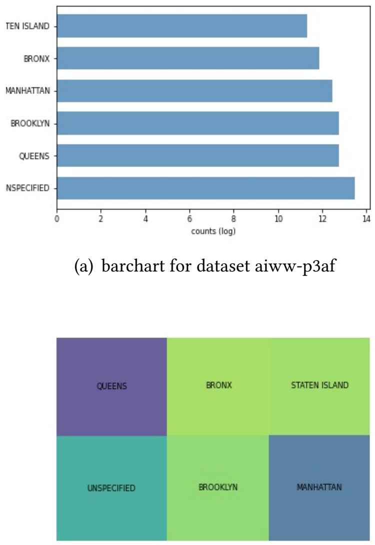
Figure 1: Visualisation of dataset 3rfa-3xsf after data clean



(a) barchart for dataset 8eq5-dtjb

(b) treemap for dataset 8eq5-dtjb

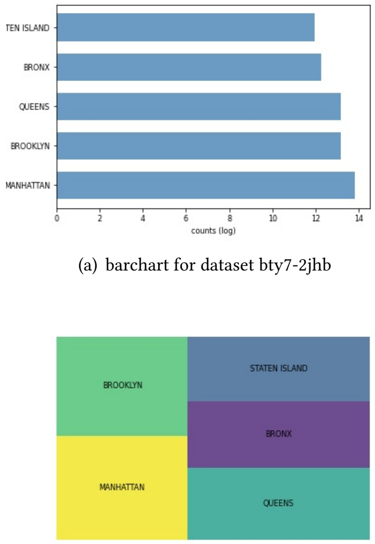
Figure 2: Visualisation of dataset 8eq5-dtjb after data clean



(a) barchart for dataset aiww-p3af

(b) treemap for dataset 8eq5-dtjb

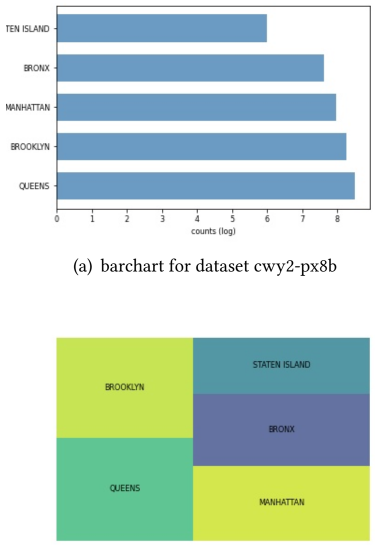
Figure 3: Visualisation of dataset aiww-p3af after data clean



(a) barchart for dataset bty7-2jhb

(b) treemap for dataset bty7-2jhb

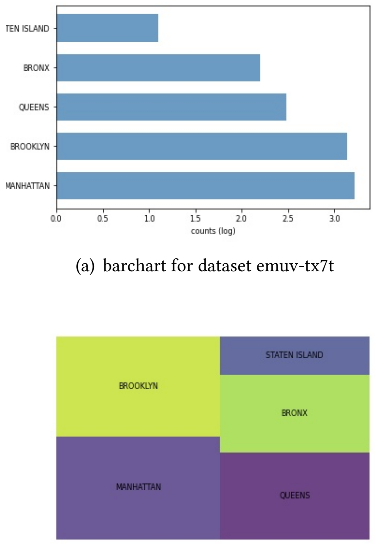
Figure 4: Visualisation of dataset bty7-2jhb after data clean



(a) barchart for dataset cwy2-px8b

(b) treemap for dataset cwy2-px8b

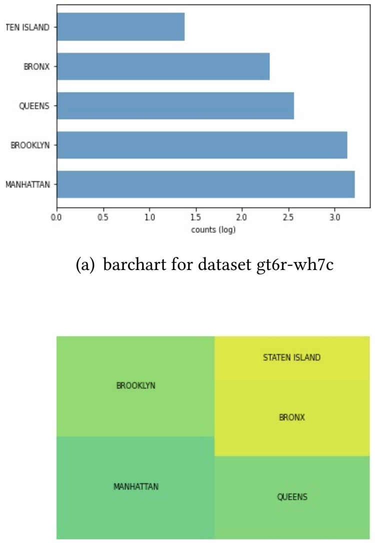
Figure 5: Visualisation of dataset cwy2-px8b after data clean



(a) barchart for dataset emuv-tx7t

(b) treemap for dataset emuv-tx7t

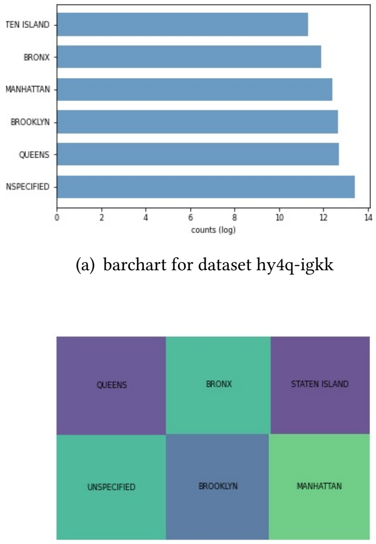
Figure 6: Visualisation of dataset emuv-tx7t after data clean



(a) barchart for dataset gt6r-wh7c

(b) treemap for dataset gt6r-wh7c

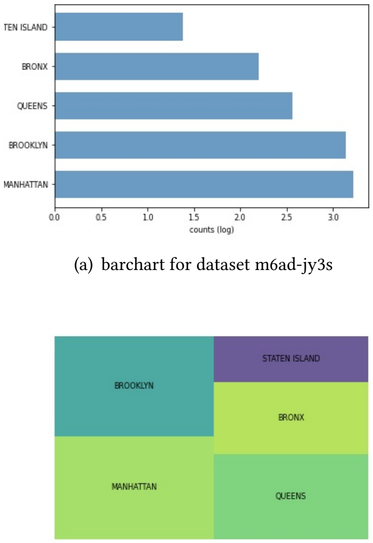
Figure 7: Visualisation of dataset gt6r-wh7c after data clean



(a) barchart for dataset hy4q-igkk

(b) treemap for dataset hy4q-igkk

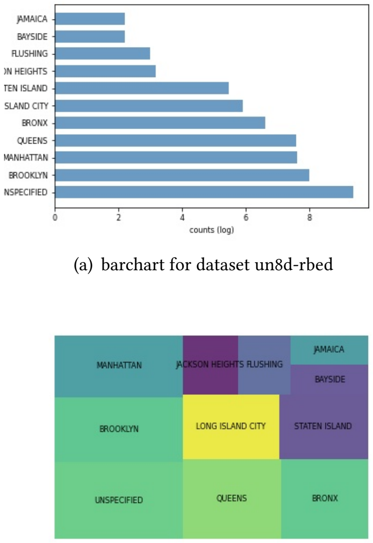
Figure 8: Visualisation of dataset hy4q-igkk after data clean



(a) barchart for dataset m6ad-jy3s

(b) treemap for dataset m6ad-jy3s

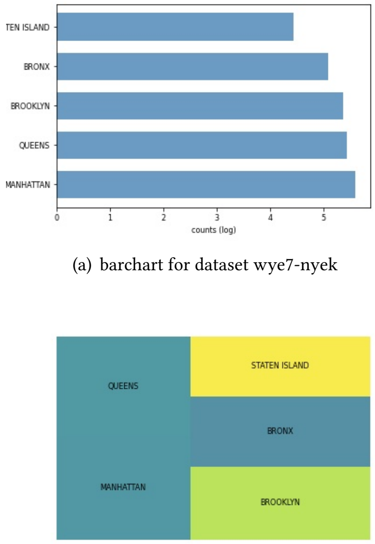
Figure 9: Visualisation of dataset m6ad-jy3s after data clean



(a) barchart for dataset un8d-rbed

(b) treemap for dataset un8d-rbed

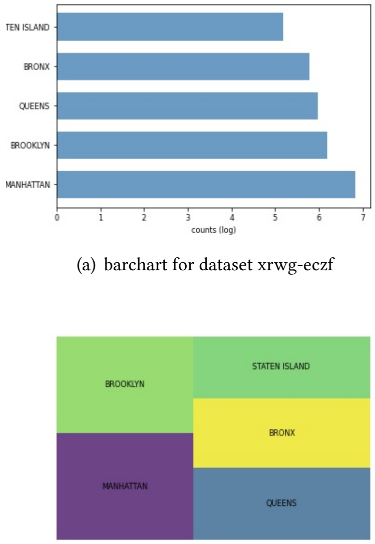
Figure 10: Visualisation of dataset un8d-rbed after data clean



(a) barchart for dataset wye7-nyek

(b) treemap for dataset wye7-nyek

Figure 11: Visualisation of dataset wye7-nyek after data clean



(a) barchart for dataset xrwg-eczf

(b) treemap for dataset xrwg-eczf

Figure 12: Visualisation of dataset xrwg-eczf after data clean

6.2.6 Challenges and limitations. Though our new strategies could clean all the rows marked as dirty data by us, there were still be some challenges and limitations for us:

- Data FY17 BID Trends Report Data, FY18 BID Trends Report Data, FY19 BID Trends Report Data, FY20 BID Trends Report Data used abbreviations to represent the borough value, though we created a reference data containing all the abbreviations we found, there might be more abbreviation used in other datasets which was not included in our reference data. Hence the reference data should be extended with more potential values.
- In some cases, the KNN cluster could not mark all the typo issues in values, hence these typo should be fixed manually sometimes. A more effective way could be used to make more precise predictions among all the data.
- For rows with combined borough values, in our case we only handled the borough values splitted by semicolons, however, there might be other datasets that combined these borough values with other symbols, which should be improved to fit all the potential datasets with this kind of problem.
- We noticed that for some borough values from our visualization, such as Jamaica and Flushing, the frequency of the data was low compared to other boroughs, which caused the unbalance of the value counts. Hence more datasets containing these low-frequency values could be used.

6.3 Date

6.3.1 Column Introduction. Date related columns are very common and may exist in the vast majority of datasets. However, in this project we need to find the columns that may overlap with the original dataset. Therefore, we used the similarity method described above to find columns with similar names to the column "Agenda Start Date" in the original dataset.

6.3.2 Our original strategy. In the original dataset, the string format of the column "Agency Start Date" is already standardized. Their years, months and days are separated by a slash character, that is to say, they follow the "%m/%d/%Y" datetime format. Therefore, the data processing for this column is quite simple, we would just split the string by the slash character to get the corresponding year, month and day. It turns out that values of years are more important than others, so our original strategy would mainly focus on them.

- First, we deleted the null values.
- Second, we split each string by slash character "/". Because it follows "%m/%d/%Y" datetime format, the third string should be year.
- We simply deleted all the values with years greater than 2021.
- Also, we used common sense to consider some years such as 1901 as outliers.

6.3.3 Result of effectiveness and problems in new datasets. When our original strategy was applied only to the original dataset, the results were seemed to be good and there was no problem with precision. Unfortunately, our original strategy actually has major problems, especially when we try to apply it to other datasets. In fact, our original strategy was simple and crude and not extensible at all. It is easy to identify many of the problems in data cleaning strategy described above:

- When parsing datetime-like strings, apparently we should not just split the year, month and day by a certain kind of delimiter. For example, the string "13/41/2020" is not a valid datetime, but it is impossible to validate by just splitting the string. Usually we should use the datetime-related methods in python module, passing in a datetime format string to parse and validate.
- Further more, there may be different datetime formats in a single dataset, which may require us to use several different formats to parse.
- It is completely wrong to delete all data greater than the current year because it does not take into account the meaning of column names at all. "Start Date" may have years greater than current year, of course. More inspection is needed.

We inspected new datasets "bty7-2jhb.csv", "ptev-4hud.csv", "un8d-rbed.csv", "wye7-nyek.csv" and noticed that dataset "bty7-2jhb.csv" had pretty chaotic format.

6.3.4 Apply our refined strategy. Our refined strategy consists of following steps:

- Try to parse datetime-like strings with more than 1 formats. For example, in this project we parsed with both "%m/%d/%Y" and "%Y-%m-%dT%H:%M:%S".
- During parsing, we will find out the invalid strings, try to extract the year, month and day, then generate figures by visualization method. (Figure 13)
- In order to identify outliers, we draw histograms for all years (Figure 14), as well as histograms for years equal or less than the current year (Figure 15) for comparison.
- If we still cannot confidently identify the outliers, then apply clustering algorithms such as DBSCAN to cluster the possible outliers. (Figure 16)

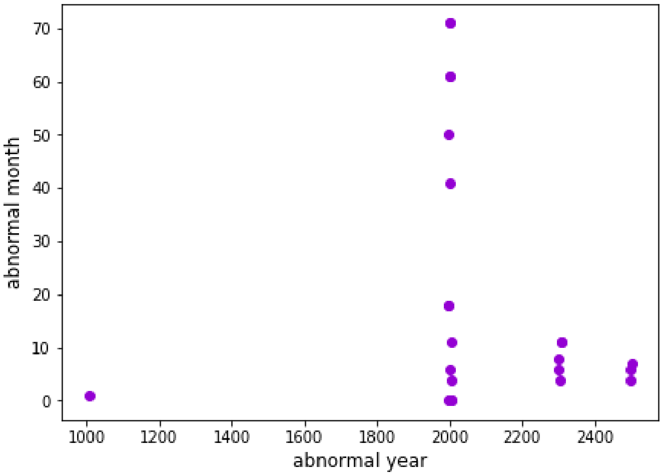


Figure 13: Visualisation of malformed datetimes obtained by data cleaning

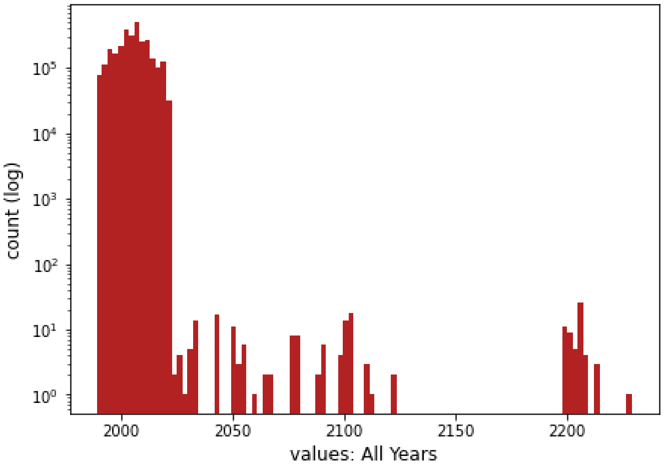


Figure 14: Histogram of value count for all years

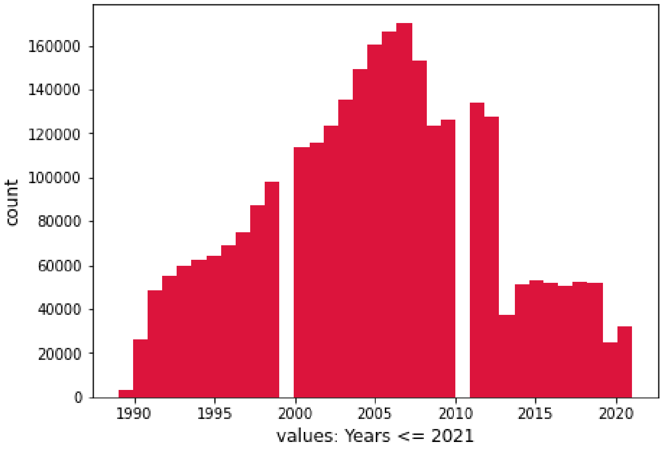


Figure 15: Histogram of value count for years <= 2021

Our strategy found 29 rows of invalid strings out of the total 2865919 rows. There are 199 rows with a year greater than the current year. But by analyzing the clustering results, we decided to define the 167 rows that greater than year 2033 as outliers.

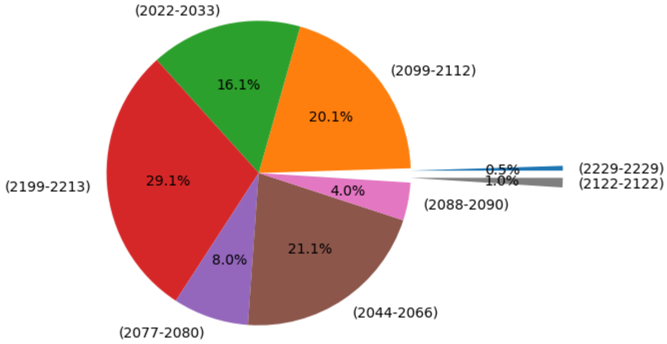


Figure 16: Clustering result for possible abnormal years obtained by data cleaning

6.4 Salary

6.4.1 Column Introduction. These kinds of columns are related to "Salary" or "Paid", usually many datasets will have more than one related column. Depending on the definition of "Salary" or "Paid" for each column, their values may be very different. Because of the variety, defining which value is correct or wrong can be difficult or somewhat ambiguous.In this project, we would mainly focus on the distribution of values.

6.4.2 Our original strategy. In the original dataset, related columns are "Base Salary", "Regular Gross Paid" and "Total OT Paid". The raw data in these columns is already relatively clean, so data profiling and data cleaning is rather simple. For example, each row in the raw data has a definite value and there are no NaN values; and its data format is either a standard integer or floating point number, there's no other special characters.
Since the data format of the original dataset doesn't have any problem, the data profiling for the original data set has following two steps:

- First, count the minimum, maximum, mean and median.
- Then, draw the histogram of frequency distribution of each column. These statistics as well as the visualization images can help us to better identify outliers.

Once we got the outliers from the data profiling, we performed the data cleaning strategy by deleting those abnormal rows. Among these, negative values were the most obvious outliers, as it is clearly impossible for a person to have a negative salary.

6.4.3 Result of effectiveness and problems in new datasets. For other types of data, the above method may find some exceptionally small or large outliers. However, there is one special feature about the data of salaries. That is, salaries vary drastically from person to person, because not only there's wide gap between the rich and the poor, but also the ways in which they are paid can be so different that many of the so-called "salary" values actually make little sense. In fact, even considering negative values as outliers is debatable. Some sources suggest that negative values might be used for correcting previous payroll data in some cases.
Overall, measuring the effectiveness of our data cleaning strategy for salary columns turns out to be tricky, so we decided to spend more effort on profiling rather than cleaning.
We inspected new datasets "8eq5-dtjb.csv", "emuv-tx7t.csv", "gt6r-wh7c.csv" ,"m6ad-jy3s.csv" and noticed that salary column of dataset "emuv-tx7t.csv" contains invalid string which can not be directly converted to integers. What's more, there're also many NaN values, which do not appear in the original dataset.

6.4.4 Apply our refined strategy. Before cleaning the invalid string from the dataset "emuv-tx7t.csv", we first applied the original strategy on other datasets which have the valid format. However, Figure 13 histogram shows that there's no negative values in these new datasets, and since we didn't remove NaN values, the original strategy didn't actually clean anything.

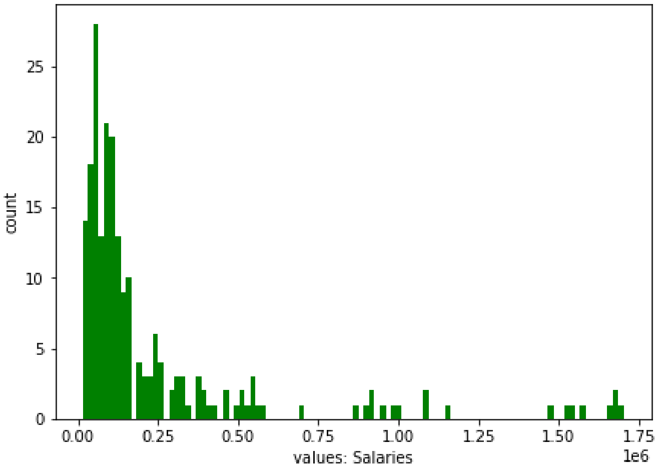


Figure 17: Visualisation of salary among all datasets with our original strategy

Here we noticed that some people earn more than 1 million (dollars?). Although these values are surprisingly large, we cannot simply consider them as outliers, because such high salaries are entirely possible.
In the dataset "emuv-tx7t.csv", there're strings like " 64,575 " or " - ", with unnecessary spaces, commas and dash characters which represent NaN. So our refined strategy will properly convert them into numbers.

- Remove spaces and commas.
- Replace dash character with NaN.
- As for the NaN values, we decided to keep them since standard IEEE 754 float format supports NaN.

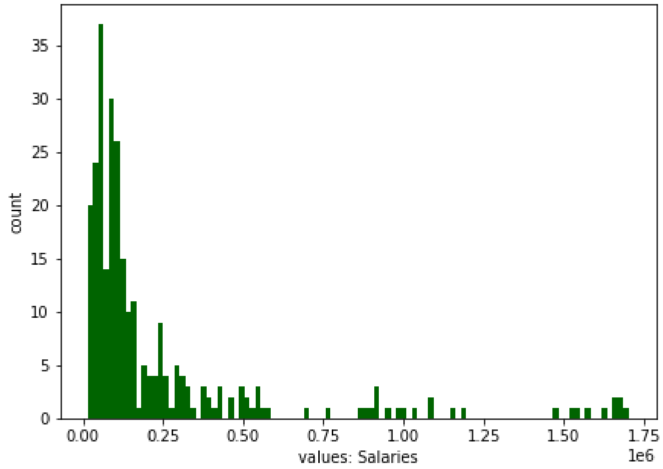


Figure 18: Visualisation of salary among all datasets with our refined strategy

7 EXTEND TO ALL THE DATASETS

The more datasets we process, the more date string formats we may encounter. Therefore, the extended strategy needs to be able to detect a variety of different formats and parse them correctly. Our refined strategy has been proved to be able to extend from single datasets to multiply, hence we can just apply those methods to any datasets with similar attributes. If we want to run our approaches on all NYC Open Data datasets, we can implement our methods in Spark or Hadoop since those two utilities are very highly scalable and can be used to solve vast and intricate data problems. And also, we can extract the overlapped columns within the list of all datasets and run our different cleaning strategies column by column in order to cover all of the datasets.

Next we would express our concern. Since our strategies would create reference data for columns with borough and agency name with small size, which meant that this strategy could also extend to all the datasets among the whole project data folder on HPC server as we extended our refined strategy to the whole data collection we handled. Two things should be emphasised:

- Since the reference data was extracted, profiled and cleaned from data collection with only dozens of number spotted out by us the reference data should be enriched and could cover wider range of datasets
- For handling larger scale of datasets, spark or hadoop should be used as we did when generating dataset abstract in section 5.1 to fit the memory required to fit the data size.

With the concern above, the procedure of extending our strategies to the whole datasets should be:

1) Inspect the larger scale of data and filtering the datasets with overlap
2) Enrich the reference to make it be able to cover all the potential datasets in the collection.
3) Transfer the code to a spark program, then run the data clean spark program on HPC server.
4) Visualisation of the results obtained and analyse the results.

8 CONCLUSION

This section we would made a summary regarding the Problem Formulation And Object Section.

- We generated a datasets abstract, measuring the similarity between our starting dataset and all the dataset from NYC OpenData by k-shingles with jaccard distance according the this reference data. Inspected the overlap datasets and handled them.
- We applied our original strategies to the datasets we found, then profiling the data cleaned by this strategy, identifying the problem existed and refined our strategy.
- Our original strategy was already able to detect most of the problems in the datasets, with high effectiveness. But with more datasets, it could still got some problem regarding the size of our reference data, and the irregularity of data recording, especially the differences of data formatting such as usage of abbreviation, or date format.
- We refined our strategy to make it be able to cover all the datasets we handles. We extended our strategy not only on the overall columns, but also each of the dataset. We inspected the effectiveness of our refined strategy supported with some visualisation techniques.
- Reference data created by was supposed to handle all the columns in other datasets which may have the same issues with the datasets we handled in the columns which were overlap with the columns we handled.
- The approaches to extend our strategies to all the datasets should enrich the reference data, and using proper tools to fit the scale of datasets.

9 REFLECTION

In the end of the paper, we would discuss some other topics beyond this project, including our extra contribution, the overall limitations and challenges of this project.

9.1 Our Contribution

Among all the overlap column handled in our project, we noticed that there were some columns we cleaned could be used as reference data. Especially for some proper nouns in New York City. Hence during this project, we took two column which may be valuable for the future usage.

We created pull request to openclean-reference-repository, with two reference data. One was the mapping of abbreviation of borough names in New York City, another was a mapping of New York City agency name and the abbreviation of department they belonged to. All the code including spark programs and jupyter notebook used for us was available on our GitHub repository: Link to GitHub Repository

All these reference data including reference of zip code - borough name was included here.

All the jupyter notebook were available here.

All the spark program were available here.

Note that all the visualisation figures were also available on the repository directory here.

All the dataset abstract generated by spark program and used by us to inspect the overlap datasets were available here.

9.2 Overall Limitation and Challenges

Though we implemented a lot of methods to clean the inappropriate data, we still faced some challenges and limitations with our approaches. Different from the limitations and challenges we mentioned regarding columns under section 6, in this we would deliver some limitations found by us for the overall project, on both tool usage and the potential issues in datasets.
For text data:

- When we were dealing with the columns with text data types (mostly names), the same data can be written in different forms and combinations which may not be included in the reference data.
- The reliability of KNN cluster can be doubtable, after we implement the KNN cluster and sometimes we still have to correct some errors by ourselves manually and this limitation might cause more problems when we are trying to deal with massive datasets in the future.
- The symbols between combined text values can be very confusing and hard to deal with since we can not understand the real meanings of those different kinds of symbols, thus we might need a better and comprehensive method to fit all of the datasets.
- Data balance problems. Within a dataset, one value can have very low or high frequency compared to other values which may cause an unbalanced issue of the value counts. We might need to use more datasets containing those low or high frequency values to fix the problem.
- Limitation of visualization because of the scale of datasets. When a single column contains way too many distinct types or kinds can make the visualization to be meaningless since the chart can be extremely large and hard to read. The scale of data can sometimes be the major limitation of data visualization.

For digital data:

- The focus of datetime-related data cleaning is on parsing strings and detecting invalid formats. Other than that, defining outliers is not difficult. The issue of date format may exist in other datasets, due to the localization, and the system used to collect datetime.
- Since for some datasets, we did not know the background of the datasets collected, such as the negative values in salary, or the year involved, sometimes it would be hard to identify the outliers regarding digital data. Such as What's the period of data collection? and Who was involved in the dataset? To handling larger scale of datasets, this kind of background would help the data scientists identify the problems in datasets more precisely.

ACKNOWLEDGMENTS

Throughout the whole jobs done of this project, we are happy to participant into this course and learnt a lot from this class. We'd like to thank our instructor Professor Juliana Freire who gave us a lot of instructions during the progression of this project.
In addition, we are aware that the completing of this project would credit to our team members, who always contribute their efforts and show their passion to the group work of the CS-GY 6513 Big Data course. We had a great and happy teamwork during this one month in the period of assignment 3 and course project.
We are also glad to meet some excellent classmates during this course, which was also one of the biggest gaining we got.

REFERENCES

[1] en.wikipedia.org. 2021. Data profiling - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Data_profiling> [Accessed 9 December 2021].
[2] www1.nyc.gov. 2021. NYC Resources | Agencies | City of New York. [online] Available at: <https://www1.nyc.gov/nyc-resources/agencies.page> [Accessed 9 December 2021].
[3] en.wikipedia.org. 2021. Reference data - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Reference_data> [Accessed 9 December 2021].
[4] City of New York, N., 2021. NYC Open Data. [online] opendata.cityofnewyork.us. Available at: <https://opendata.cityofnewyork.us/> [Accessed 9 December 2021].
[5] en.wikipedia.org. 2021. Apache Spark - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Apache_Spark> [Accessed 9 December 2021].
[6] GitHub. 2021. GitHub - VIDA-NYU/openclean: openclean - Data Cleaning and data profiling library for Python. [online] Available at: <https://github.com/VIDA-NYU/openclean> [Accessed 9 December 2021].
[7] GitHub. 2021. Big_Data_Project/assignment_3_final.ipynb at main · NickNameInvalid/Big_Data_Project. [online] Available at: <https://github.com/NickNameInvalid/Big_Data_Project/blob/main/assignment_3_final.ipynb> [Accessed 9 December 2021].
[8] data.cityofnewyork.us. 2021. [online] Available at: <https://data.cityofnewyork.us/City-Government/Citywide-Payroll-Data-Fiscal-Year-/k397-673e> [Accessed 9 December 2021].
[9] GitHub. 2021. nyc-housing/nyc-zip-codes.csv at master · erikgregorywebb/nyc-housing. [online] Available at: <https://github.com/erikgregorywebb/nyc-housing/blob/master/Data/nyc-zip-codes.csv> [Accessed 9 December 2021].