Data Quality Index - An Open Source Spark Framework

SRIPHANI BELLAMKONDA, New York University, USA
KAARTIKEYA PANJWANI, New York University, USA
SUMEDH SANDEEP PARVATIKAR, New York University, USA

1 ABSTRACT

This research focuses on the imperative task of Data Quality Assessment (DQA) in the era of Big Data. As data's role in analytics and decision-making intensifies, the paper emphasizes the necessity of ensuring data quality. Traditional data cleaning processes are resource-intensive, and with the proliferation of diverse datasets from various sources, a more efficient and comprehensive approach to DQA becomes essential while allowing for a domain adaptable semi-automated approach. The paper proposes dimensional metrics, including Uniqueness and Completeness for assessing generic dataset quality. These metrics, integrated into a data quality score, offer users a succinct overview of the usability of Datasets. Acknowledging the subjectivity of data quality to user needs, the research aims to expand these dimensions further and additionally allows the user to write domain agnostic data quality rules. To tackle the challenge of processing large datasets efficiently, the paper advocates leveraging the Spark Framework (pyspark) for computation, providing a scalable solution to contemporary DQA complexities. Finally, the paper uses four real world datasets from NYC OpenData to experiment with the DQA and makes the code available at https://github.com/soulpower007/DataQualityFramework

ACM Reference Format:
Sriphani Bellamkonda, Kaartikeya Panjwani, and Sumedh Sandeep Parvatikar. 2018. Data Quality Index - An Open Source Spark Framework. 1, 1 (December 2018), 11 pages. https://doi.org/XXXXXXX.XXXXXXX

2 INTRODUCTION

Undoubtedly, we are in the age of Big Data defined by the 3Vs (Volume, Velocity, Variety). This vast pool of data is crucial for analytics and making informed decisions. Its applications extend beyond large institutions, playing a crucial role in training deep learning models which continuously improve and learn from the data. For example, with the Generative Pretrained Transformer-3 (GPT-3) model, over a billion parameters were trained on hundreds of gigabytes of data, utilizing the computational power of hundreds of NVIDIA GPUs. It is also not untrue to assert that an increased volume of data can significantly enhance the performance and insights derived from today's major applications. Just as during the Industrial Revolution, fuel propelled automobiles to unexplored destinations, data can propel innovations to achieve unprecedented advancements. However, this analogy is imprecise. Despite the proliferation of data sources facilitated by sensors and other devices or tools, most of that data is unfiltered and tainted. Datasets are also filled with data in unrecognizable formats, and cannot be used directly. Subsequently, If the data is misinterpreted, the respective

Authors' addresses: Sriphani Bellamkonda, sb7179@nyu.edu, New York University, 6 MetroTech Center, Brooklyn, New York City, New York, USA, 11201; Kaartikeya Panjwani, New York University, 6 MetroTech Center, Brooklyn, New York City, New York, USA, kp3291@nyu.edu; Sumedh Sandeep Parvatikar, New York University, 6 MetroTech Center, Brooklyn, New York City, New York, USA.

analysis would also be flawed. Hence it is not illogical to say that data quality is as important as the algorithms used to derive insights from it.

Traditionally, we clean the data before processing or modelling which takes a major chunk of the resources until we're completely satisfied with the dataset. In the modern day, we have tons of different datasets from different sources, and we have the capability to also augment the conventional dataset with new features joined using specific columns. With all these capabilities, it's a very tedious task to clean every dataset before learning that it is not useful or meaningful for our use case. **Data Quality Assessment** is essential to give necessary information to the user about the dataset like the existence of null values, row duplication, different formatting errors, outliers and many more. This is to inform the user so that he or she could make a sound decision of choosing a dataset for their purpose. But data quality is subjective to a user's requirements and their use case. With many years of research [6], there are some defined dimensions for assessing the data quality like Accuracy, Completeness, Consistency, Timeliness and more; however, they are based on the knowledge of a particular domain. Hence this necessitates the use of a semi-automated approcah to Data Quality. In this paper, we will introduce an opensource distributed framework on pyspark and propose a set dimensional metrics to define the data quality for datasets which might help the users to get a high-level overview of the dataset via the data quality score. Moreover, we also provide semi-automated data quality metrics which can be easily defined by the domain expert such as Data format, Data Range, and Master Data to name a few. Some of the fully generic dimensional metrics defined are Uniqueness and Completeness. More detailed information would be provided about these dimensions in the Methodology section. We hope to add more dimensions as we proceed to capture the essence of the dataset quality using a score. To calculate data quality scores on huge datasets is time consuming, especially when there are many dimensional attributes. And as time continues, we will have larger datasets thus piling up on existing issues and forcing to increase the compute power to calculate the quality scores and further processing. A suitable approach to resolve such issues would be to leverage current distributed systems which are capable of handling huge datasets efficiently. Hence we will be building our library on top of Spark Framework (pyspark) to tactfully take advantage of its framework and Hadoop Distributed File system in processing the data efficiently.

3 RELATED WORKS

Research on Data Quality Assessment dates to the early 2000s where Pipino et al [5] explains different techniques to assess the quality of the data objectively or subjectively. Data quality is of utmost importance to businesses, research teams, governments, etc. As the data becomes the basis of shaping new decisions, policies and more. Pipino et al explore different data dimension measures like Accessibility, Completeness, Interpretability, Objectivity, Relevance and more. Further, it is also mentioned how some of these dimensions can be objectively and subjectively quantified using three functional forms (Simple Ratio, Min, Max Operations). Pipino et al [5] finally concludes stating that there are no generic data quality metric measures, and one could combine subjective and objective data quality measures as demonstrated in the paper To evaluate and assess the data quality issues, Kelly et al [4] used multivariate outlier detection techniques. This was an application within the ONDRI project, where trying to validate and find data errors was becoming a tedious task. Moreover, the margin of error still existed even after spending resources in identifying the errors in the data. To avoid errors in an efficient and elegant manner in today's ever-growing datasets, the authors came up with an approach to use outliers to detect patterns for errors. In this paper, they've used two multivariate outlier detection algorithms Minimum covariance determinant (MCD) and Robust Principal Component Analysis (RPCA). In addition, they've also used univariate algorithms like univariate MCD and Boxplots and compared them with the former. Kelly et al were successfully able to identify the errors using the outlier algorithms but noticed some limitations with the approach such as if there are enough errors in the dataset and there is no pattern observed, then it would be difficult to detect the errors in the dataset. These algorithms also don't work well with categorical variables and further research needs to be done to quantify them. This research introduces a comprehensive objective quality framework for Linked Open Data (LOD), building upon previous efforts and emphasizing objective data quality measures. Assaf et al [1] identify and categorize 64 quality indicators related to entities, datasets, links, and models. The paper also surveys over 30 tools that measure various quality aspects of LOD, revealing gaps and highlighting the need for a comprehensive evaluation framework, particularly for dataset-level quality assessment. To address this, an extension of Roomba, a tool for assessing and generating dataset profiles, is presented, covering 82% of the suggested objective quality indicators [2]. The experiments conducted using Roomba on the LOD cloud showcase a concerning state of datasets, with most exhibiting low scores in completeness, provenance, licensing, and comprehensibility. However, the paper lacks explicit discussion on the subjectivity involved in selecting and categorizing quality indicators, leaving room for differing interpretations. Additionally, there is a need for more comprehensive evaluation metrics or benchmarks to assess the performance of the proposed framework against existing solutions. The scalability of the tool, although slated for evaluation on larger data portals, lacks specific details or preliminary assessments. Furthermore, ethical considerations related to data quality assessment, such as privacy concerns and potential biases, are not thoroughly addressed.

In future work, the authors plan to enhance Roomba by integrating tools assessing model quality and syntactic checkers, aiming for complete coverage of proposed quality indicators. They also intend to suggest ranked quality indicators to enhance the quality report and plan to run the tool on CKAN-based data portals, providing periodic reports to monitor quality evolution. The future roadmap includes extending the tool for other data portal types such as DKAN and Socrata and evaluating its scalability on larger data portals. Cai et al [3] propose a dynamic data quality assessment process with a feedback mechanism tailored to the unique characteristics of big data. The process defining the goals of data collection based on strategic objectives or business requirements. The selection of data quality elements varies across different business environments, and the paper highlights the importance of adapting assessment criteria to the specific characteristics of data sources. Data cleaning is introduced as a crucial step to detect and remove errors and inconsistencies, contributing to overall data quality improvement. The paper distinguishes between qualitative and quantitative methods for data quality assessment, with qualitative methods relying on expert analysis and quantitative methods providing a more formal and objective approach. The results of the assessment are compared with the established baseline, and if the data quality falls short, new data acquisition is recommended.

4 PROBLEM FORMULATION

The problem that this study addresses is a need for quickly and efficiently finding a data quality score of a dataset with adaptable human intervention. This is done in order to quickly assess the usability of a dataset before undergoing the time intensive process of data cleaning. As the persistence of not having a usable semi automated data quality score prolongs the duration of data analytics projects, this solution is of consequence and needs to be aptly delivered. This kind of study also requires the development of a framework in pyspark which facilitates the algorithms and dimensions mentioned in this paper. We aim to go through an emprical analysis of four dataset which have been uploaded to the Hadoop Distributed File System in NYU's dataproc.

Our Data Quality Index with the respective dimensions will follow:

$$\mathrm{dqa} = \frac{1}{n}\sum_{i=1}^{n} w_i \cdot \mathrm{dim}_i$$

- dqa is the weighted Data quality index.
- n is the total number of dimensions.
- $\sum_{i=1}^{n}$ denotes the summation over all dimensions.
- w_i is the weight associated with dimension i.
- dim_i is the individual score of dimension i.

To further reiterate, our objectives for this work are:

- Identify and define relevant data quality dimensions that are particularly pertinent to the challenges posed by Big Data.
- Make the framework openly available for usage and development by the masses.
- Efficiently compute the metrics by making use of parallelism and a distributed environment.
- Offer flexibility and adaptability to change the index to reflect the aspects more aligned to the individuals use-case.
- Emprically test the framework in computing the data quality scores of multiple datasets to validate the solution.

5 OUR METHODOLOGY

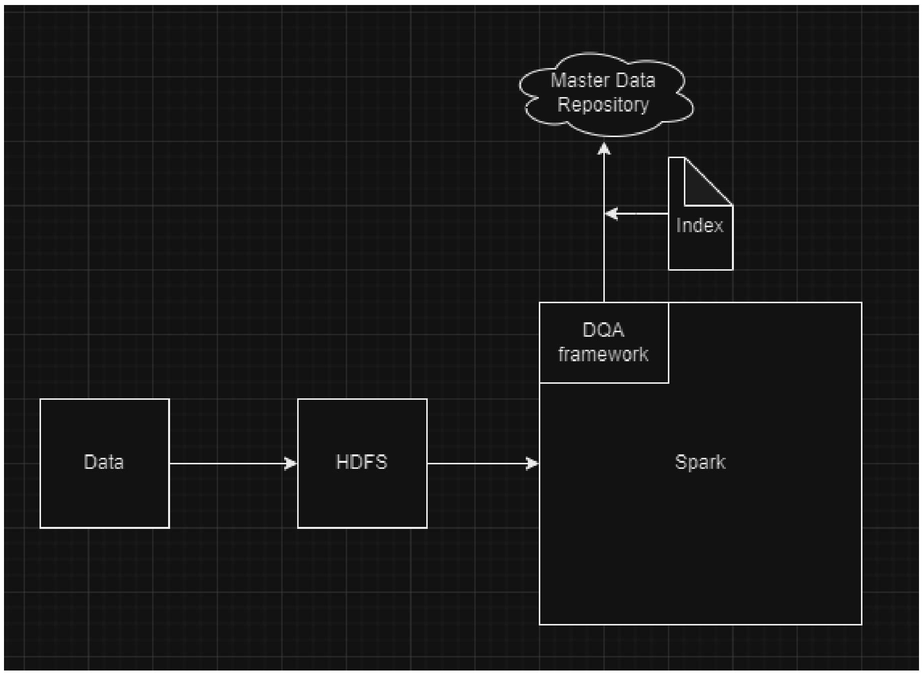

Fig. 1. Your Caption Here

We aim to create a semi-automated Data Quality Index which readily captures the intrinsic metrics of a given dataset while allowing the user to easily define the extrinsic qualities of the columns in the dataset specific to the application or use-case. One key merit to our approach is that we utilize Spark to create a data quality library and in doing so utilize parallelization seamlessly and allow anyone to use it with a simple install and import. The dataset should just be uploaded to the hadoop distributed file system and we will capitalize on its advantages to draw insights faster. Second key merit of our approach is that it is open source, hence allowing contributions en masse for rapid development. Let's first define our intrinsic data dimensions, followed by extrinsic data dimensions, and see how we can combine the above to portray a data quality index.

5.1 Uniqueness

Uniqueness is important as having at least one column as a unique identifier of every row in a dataset tells that a dataset is more standard. Most datasets we encounter, however, do have unique identifying columns. Hence this metric can be used to single out those datasets which don't have one.

$$Uniq_{Col_i} = \frac{\text{No. of distinct records for column i}}{\text{total records}} \tag{1}$$

$$Uniqueness = \left\lfloor Max\left\{Uniq_{Col_1}, Uniq_{Col_2}, ..., Uniq_{Col_n}\right\}\right\rfloor \tag{2}$$

5.2 Completeness

Regardless of the type of data, if the data element is missing then we can do nothing about it. Hence it is essential to capture how many of the rows in the columns of a dataset actually have values in them. We've also implemented it as an extrinsic data quality where the user can provide certain values for the columns to be treated as null values which would give the user more power in understanding the usefulness of the data.

$$Comp_{Col_i} = \frac{\text{No of non Null Records}}{\text{Total Records}} \tag{3}$$

$$Completeness = \frac{\sum_{i=1}^{n} Comp_{Col_i}}{n} \tag{4}$$

5.3 Duplicate Records

We consider a set of identifying columns as candidate columns for checking the Duplicate records. We parse them and concatenate all the values in a row and then we hash this concatenated string using Sha 256 to generate a smaller fingerprint. After this the task of data duplication can be reduced to the standard word count problem with this concatenated and hashed value as the key. This is can be then easily retrieved with the countDistinct function as part of pyspark. However, we also provide another method to allow for faster computation with 1% approximation error. This error is well within our accepted threshold since this is only to provide a DQA index which will signal usability of a dataset. Hence, for this we utilize a cardinality estimation algorithm called the HyperLogLog algorithm.

$$Fingerprint_K ey = H(Col_1|Col_2|...|Col_m) \tag{5}$$

The HyperLogLog algorithm estimates the cardinality of a multiset by employing a hashing mechanism and the harmonic mean. This algorithm takes advantage of the fact that when numbers are spread uniformly across a range, the count of distinct elements can be approximated from the largest number of leading zeros in the binary representation of the numbers. Suppose k is the number of leading zeros in the binary representation, then a crude estimation of 2^k elements can be made, which is further optimized with the LogLog sketching algorithm. The HyperLogLog algorithm can be formalized with two main components: the raw estimate and the bias correction.

(1) **Raw Estimate:**

The raw estimate is calculated as

$$E = \alpha \cdot m^2 / Z$$

where:

- E is the raw estimate of cardinality.
- α is a constant factor (specific to the algorithm) that is chosen based on the precision required. Commonly used values are 0.7213 for 64 registers and 0.709 for 32 registers.
- m is the number of registers in the HyperLogLog data structure.
- Z is the harmonic mean of the register values.

(2) **Harmonic Mean (Z):**

The harmonic mean Z is calculated as

$$Z = \frac{1}{\sum_{i=1}^{m} 2^{-M[i]}}$$

where $M[i]$ is the value stored in the i-th register.

(3) **Bias Correction:**

A bias correction is applied to the raw estimate to account for systematic errors, especially in the small cardinality range. The bias correction formula is

$$E_{\text{corrected}} = \begin{cases} E, & \text{if } E \leq \frac{5}{2} \cdot m, \\ -m \cdot \ln(1 - \frac{E}{m}), & \text{otherwise.} \end{cases}$$

5.4 Range Adherence

Range Adherence is a vital part of data quality as it defines whether the data exists in a particular data range. We consider that Range Adherence can be an extrinsic as well as intrinsic data quality dimension. As part of our implementation, we've completed the extrinsic property of the dimension and the end user can define the range for a particular or multiple columns of interest. This is quite useful for detecting application agnostic data errors. For example a generic Age column might have an integer data type. However, the range could differ based on the type of study. For instance, for humans it would be 0-100 and for animals it would be different. This can be captured by specifying the range or by detecting outliers assuming the majority of records fall within the right Age range. As part of intrinsic data quality dimension, we can validate fundamental properties of certain attributes like distance (cannot be negative), temperature (cannot be below 0K) and many more. Identifying the type of attribute and the unit of the attribute are of essence in solving the problem and can be taken up as part of future work.

$$RAdh_{Col_i} = \frac{\text{No of records in Range}}{\text{Total Records}} \tag{6}$$

$$Adherence = \frac{\sum_{i=1}^{n} RAdh_{Col_i}}{n} \tag{7}$$

5.5 Format Adherence

Format Adherence is a data quality problem when we have the data from multiple sources in different formats. We can have the same data in different formats thus complicating further processing. As part of our framework implementation, we propose to support the extrinsic dimension property of data quality and based on the user's study requirements, one can define the regex format the data must follow. The user can define the patterns for each column in a YAML file which is consumed by our framework in validating the format adherence.

This dimension provides user with a powerful tool to not only check for format adherence but also validate other aspects of the data.

$$FAdh_{Col_i} = \frac{\text{No of records following the format}}{\text{Total Records}} \tag{8}$$

$$Adherence = \frac{\sum_{i=1}^{n} FAdh_{Col_i}}{n} \tag{9}$$

5.6 Functional Dependency Adherence

The dependency between any two columns can be defined based on the use case and if this constraint is important to the end user to be maintained. We take in a transformation given by the user such that Col2 = F(Col1). We enforce it across all the rows in Col2 and determine which ones do not meet the requirement and diminish the quality factor respectively.

5.7 Master Data Adherence

Master Data or Reference Data can be passed down for the respective columns, and each row from that column can be checked to determine whether it belongs to that reference. This is another dimension which will be taken into consideration if specified by the user. The user may upload their custom Master data in the form of a csv file. And additionally provide any tags to it which corresponds to its content.This csv will be parsed and used as the reference. We also house a repertoire of Master Data in a public AWS S3 bucket containing several master data files. These are indexed according to their tags for faster search and load in the following way such as:

```
{
    "states":["USStates"],
    "carplates" : ["USCarPlates", "IndianCarPlates"]
}
```

This indexing allows for efficient **Master Data Search**. For now we have kept it in a json file due only minor loads but this can be easily indexed with AWS Elasticsearch heavy loads. The master data search returns a set of master data files which adhere to the intersection or unions of the tag wise search requested. After browsing through the master data the user can choose one of the datafiles and load it to be used as a reference data as well for adherence checking.

5.8 Outlier Detection

In big data, outlier detection is essential for a number of reasons. First of all, abnormalities or inaccuracies in the data may be signs of problems with data quality that could influence analysis and judgment. Second, outliers have a substantial effect on statistical measures in large data sets, which may bias the results. Thirdly, by preventing aberrant data points from unduly influencing machine learning models, the identification of outliers helps improve the models' accuracy and dependability. Lastly, in big data analytics, outlier detection aids in efficient fraud protection, anomaly detection, and pattern recognition that may be masked by outliers. We have used the Interquartile Range method to identify outliers. The interquartile range is a measure of statistical dispersion, defined as the range between the first quartile (Q1) and the third quartile (Q3) of a dataset. Outliers are then identified as values that fall below Q1 - k * IQR or above Q3 + k * IQR, where k is a constant multiplier that determines the sensitivity to outliers. We have calculated the outlier scores for each numeric column of the datasets. The outlier score is a measure of the amount of outlier values present in the column. It is calculated as:

$$Q1 = \text{median}(\text{lower half})$$

$$Q3 = \text{median}(\text{upper half})$$

$$IQR = Q3 - Q1$$

$$\text{Lower Bound} = Q1 - k \cdot IQR \tag{10}$$

$$\text{Upper Bound} = Q3 + k \cdot IQR \tag{11}$$

$$OutlierScore_{Col_i} = 1 - \frac{\text{No. of outliers in column}}{\text{Total Records in column}} \tag{12}$$

5.9 Further Automations

By utilizing a few smart techniques we hope to further automate the above described approach. For example without any interference from the user, we can automatically detect any possible functional dependencies by finding inter-correlated columns. Next, we can suggest relevant master data based on the column name and respective tags which are correlated to the column. This is feasible if along with the master data we store its metadata in the form of tags for comparison purposes. Similarly, we can hypothesize the range for each of the columns by applying outlier detection algorithms and considering the minimum and maximum value of the cluster to be the approximate range.

6 RESULTS

We have currently implemented the **Uniqueness**, **Completeness**, **Outlier Detection**, **Master Data Adherence Duplicate Records**, **Range Adherence** and **Format Adherence** data quality dimensions on four data-sets, namely *Electrical Permits* Dataset, *Hire Vehicles* Dataset, *311 Sampled* Dataset, and *Parking Violations* Dataset. We can infer interesting thoughts with the data quality scores. We discuss them for each dataset in subsections below. The results are shown in Table 1. In our current implementation, we are using equi-weighted average to calculate the DQA index as we are not assuming any specific data quality dimension priority. The datasets used for assessing the data quality dimensions were downloaded from NYC Open Data Portal.

We allow the users to provide the input data to the framework through YAML file for various columns. As the YAML file is optional, the framework would then automatically only calculate the intrinsic dimensions and ignore other dimensions.

For the detection of outliers, we have detected them using the InterQuartile Range method and have assigned an outlier score to each numeric column of the datasets which lies between 0 and 1, 1 indicating that there are no outliers present in column. We have selected numeric columns from each data, for eg. in parking violations, the numeric columns are 'summons_number', 'violation_code', 'violation_precinct', 'issuer_code', 'issuer_precinct', 'vehicle_year'. For the final dataset outlier score, we take an average of the scores obtained for each column in the dataset.

Table 1. Data Evaluation Metrics for Datasets

Metrics	**Parking Violation**	**Hire Vehicles**	**311 Samples**	**Electric Permit**
Uniqueness	1	1	0	1
Completeness	0.897	0.717	1	0.778
Format Adherence	0.933	0.520	0.5	1
Range Adherence	0.42	-	0.67	0.936
Master Data Adherence	0.89	-	0.87	-
Duplicate Records	0.99	1	1	1
Duplicate Records Approximate	0.97	0.99	1	0.98
Outlier Detection	0.89	-	0.98	0.90
DQA	0.860	0.754	0.787	0.870

6.1 Electrical Permits

For **Format Adherence** dimensional data quality, we observe that we received a score of **1** indicating that the format provided in the YAML file by the user satisfies all the records for that particular column in the dataset. The Format Adherence data quality was applied on the Work_ID column with the following

```
{
    "Work_ID": '([aA-zZ0-9]+)-([aA-zZ0-9]+)-([aA-zZ0-9]+)-([aA-zZ0-9]+)-([aA-zZ0-9]+)'
}
```

which states that the key should have at least 4 hyphens in their record to be considered as a Work ID. This particular functionality can also be expanded to other string patterns involving numbers, special characters and more. For the Range Adherence data quality, we got a score of around **0.936** which states that for all the ranges provided for different columns, majority of the records adhere to it. However, this data set didn't have any identifiable reference data so we are opting out of it. We've found outliers in the numeric columns of Electrical Permits, that are 'Floor_ID', 'Sign_ID', 'Quantity', 'Cost', 'Fee_Amount', 'Fixture', 'Receptacles_AC', 'Receptacles_ATT', 'Switches', 'Outlets', 'Motors Generators', 'Total HPKW', 'Heaters', 'Total_KW', 'Transformer', 'Total_KVA', 'Sign_Square_Footage', 'Total_Circuits', 'Total_Lamps', 'Watts_Lamps', 'Sign_Transformers', 'VA_Transformer', 'Total_Watts_VA', 'Sockets_Per_Circuit'. The columns 'Receptacles_ATT' and 'Switches' have low outlier scores, and hence need more cleaning.

6.2 Hire Vehicles

In the Hire Vehicles dataset, there were a lot of patterns observed and we were able to use regex patterns to multiple columns. We received an aggregate score of **0.520** for all the columns. After closer investigation for each column, we can observe that for certain columns the score was very low implying that the data was not in the correct format to be very useful. Some of the regex expressions provided in the YAML file are the following

```
{
    'Website': '[wW]{3}.([a-zA-Z0-9])+.[cC][oO][mM]'
    'Hack Up Date': '[0-1][0-9]/[0-3][0-9]/[0-9]{4}'
    'Vehicle Year': '[0-9]{4}'
}
```

Moreover, we were also able to draw insights based on other dimensional qualities like **Completeness** to get a good understanding of whether appropriate usable data exists in the dataset. On the other hand, as the user did not provide any user inputs for **Range Adherence**, it ignores that dimension and displays that there are no scores available for any columns. This dataset doesn't have many numeric columns hence interquartile range method is not effectiive for calculating outliers in this case.

6.3 311 sampled dataset

The 311 sampled dataset was a clean dataset to understand the data quality score significance from the NYC Open Data. In this dataset, we observe that the dataset had a **Completeness** score of **1** for all the columns in the dataset, suggesting that there are no *null, white space, or other irrelevant values*. For format adherence, it had a high score of **1** for the *Zip Code* attribute. We've detected outliers on columns 'ZIPCODE', 'POPULATION', 'AREA', 'ST_FIPS', 'CTY_FIPS', 'X Coordinate (State Plane)', 'Y Coordinate (State Plane)', 'Latitude', 'Longitude'. All columns have an excellent outlier score with the average score of the dataset being 0.98, indicating the dataset has been cleaned properly and can be further used for data-driven analysis.

6.4 Parking Violation

In the Parking Violation dataset, we observed a **Format Adherence** data quality score of **0.93** and for *Range Adherence* we observed a value of **0.42**. A good score was observed for **Master Data Adherence** data quality of **0.89**. We've detected outliers on columns *'summons_number', violation_code, 'violation_precinct', 'issuer_code', 'issuer_precinct', 'vehicle_year'*. Some columns have low outlier score, bringing down the average to 0.89. Hence, this dataset need further cleaning. Finally, looking at the master data, we have references for registration state which was uploaded to our AWS S3 bucket. We referenced that data and used it for checking the registration_state values and found out that this method can help us identify incomplete values too.

Hence, we have shown that with a quick function call and a yaml file we can efficiently go through any dataset and determine its usability and compare it with other similar datasets using the data dimensions important to the user and the use-case.

7 CHALLENGES & FUTURE WORK

Few more dimensions such as Functional Dependency and Misspellings can be incorporated into this framework to make it more robust. Misspelling can be identified with the pyspellchecker library and functional dependency can be specified in two ways. First is a one-one mapping between the two columns such that a value from column a must correspond with a value from column b. Second is by specifying a function such that a transformation on column A should be present in column B. The challenge in this ,however, is to create a simple transformation language agnostic to our framework and create a parser to translate those commands into code in case of specifying through a yaml file. Conveniently, a function object can also be passed if done programmatically to the framework.

REFERENCES

[1] Assaf A., Senart A., and Troncy R. 2016. Towards an objective assessment framework for linked data quality: Enriching dataset profiles with quality indicators. International Journal on Semantic Web and Information Systems. *IJSWIS* 12, 3 (Jan.-March 2016), 111–133.

[2] Ziawasch Abedjan, Xu Chul Dong Deng, Raul Castro Fernandez, Ihab F. Ilyasl Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment* 9.

[3] Cai L. and Zhu Y. 2015. The challenges of data quality and data quality assessment in the big data era. *Data science journal* 14 (2015).

[4] Sunderland K. M., Beaton D., Fraser J., Kwan D., McLaughlin P. M., Montero-Odasso, M., and Binns M. A. 2019. The utility of multivariate outlier detection techniques for data quality evaluation in large studies: an application within the ONDRI project. *BMC medical research methodology* 19 (April 2019).

[5] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. 2002. Data Quality Assessment. *Commun. ACM* 45, 4 (Jan. 2002), 211–218.

[6] Shazia Sadiq, Juliana Freire, Renée J. Miller, Tamraparni Dasu, Ihab F. Ilyas, Felix Naumann, Divesh Srivastava, Xin Luna Dong, Sebastian Link, and Xiaofang Zhou. 2018. Data quality: The role of empiricism. *ACM SIGMOD Record 46(4)*, 35-43 pages.