Discovering Nulls and Outliers

Project Milestone Report (DS-GA 1004)

Diogo Mesquita
New York University
dam740@nyu.edu

Mihir Rana
New York University
ranamihir@nyu.edu

Kenil Tanna
New York University
kenil@nyu.edu

ACM Reference Format:
Diogo Mesquita, Mihir Rana, and Kenil Tanna. 2018. Discovering Nulls and Outliers: Project Milestone Report (DS-GA 1004). New York, NY, USA. https://github.com/ranamihir/big_data_project

1 INTRODUCTION

Missing value and outlier detection are two of the most essential, time-intensive, and subjective tasks in any form of data preprocessing. In most unstructured data sets, missing values tend to not have strictly formatted representations, and instead appear as potentially valid data values, which are known as "disguised missing data" [4][5]. This inconsistency arises due to several reasons – non-standardized ways of data collection and storage across various sources, changes in the way an organization stores null values, human error, etc. In the past, several methods have emerged to process missing values; in many cases, especially in a machine learning framework, it is typical to remove rows containing them if the total number of such rows is small, to impute them using several basic (such as mean, median, mode imputation) or other sophisticated techniques, or build entire predictive algorithms to impute them before the main task at hand is tackled [1], if their total count is relatively large.

For outlier detection, typically, the data points are modeled using a statistical distribution [7][8], and the data points that do show anomalous behavior [2] or the distribution of the postulated model by a "substantial" margin are considered outliers. One of the main drawbacks with this approach is that we might not have enough prior information to hypothesize an underlying data distribution, especially in extreme cases when the prior chosen is strongly biased.

Before null values or outliers can be explored, an essential first step is to clean and standardize the data sets by bringing them to a common format across columns and data sets. The first step would be to clean and standardize the data sets by bringing them to a common format across columns and data sets. This is especially becoming of paramount importance in today's day and age of an incessant barrage of data, which is often unstructured, raw, or too disorderly, and requires constant cleaning and processing. Unless it is brought to standardized format, identifying and analyzing what is important, discarding what is not, and strategizing by piecing all this together to make informed decisions, is bound to become a daunting challenge.

In this study, we consider a data set collection of 50 data sets from NYC OpenData. Since the overall objective of this study is only discovery, we will, as such, directly report and remove nulls and outliers.

Big Data (Spring 2018), New York University, New York

https://github.com/ranamihir/big_data_project

2 PROBLEM FORMULATION

As mentioned earlier, we will perform null and outlier detection on 50 sets obtained from NYC OpenData. The problem is broken down into three main components:

2.1 Data Cleaning

In the scope of this study, data cleaning or preprocessing pertains to parsing the data to bring to the correct format. An example of this type, specific type of data being recorded, is the case where income is recorded as "$0.00", leading it to be considered as a string object. Similarly, numeric quantities may also be recorded with commas, like "120,000", which may lead to erroneous analyses, and symbols such as these should be removed at the very first stage. One should also be mindful of zip-codes, which would be typically of numeric type by default, but should be treated as strings, specifically, conforming to only certain valid zip codes.

2.2 Missing Values

Missing values, or nulls, present in varying forms for each data set, and in many cases, also do not conform to strict format. For example, if a column is of numeric type, it may encoded as a string within quotes in many cases, such as "100.00". This occurrence becomes more common when null values pertaining to certain rows are encoded as strings, like "None", "N/A", etc., leading to the entire column being parsed as a string object. In many cases, nulls also present as values that are invalid, such as an income being negative, a point being drastically out of the range of the valid values, a probability or a percentage being negative or greater than 1, etc. Values such as "999", "-999", etc. are also typical. These will be reported and then removed for further analysis.

2.3 Outliers

The definition of outliers is subjective to the problem at hand, and has taken many forms over the years. One way is characterize them was described above, of postulating a distribution for the data points and finding points that stick out. Another approach is to use distance-based methods (such KMeans, Quartiles, etc.) and find points that have distances to centers larger than a threshold or clusters that have very few points in them [7]. In this study, we'll report all such outliers.

3 RELATED WORKS AND REFERENCES

[2] describe many parametric and non-parametric methods for the purpose of outlier detection. For parametric methods the common methods are IQR detection, Tukey's method of boxplots, z-score and the standard deviation are some of the common methods used for identifying outliers, since they assume that the data distribution

follows a specific target distribution. Although it is easy to calculate errors by this method, one major flaw with this is the assumption that real-world data follows a particular distribution while it may not be the case.

One of the most common non-parametric methods is the nearest neighbor method, which is giving a score to each sample based on its neighbor which is useful in finding global outliers. The notion of global outliers is that they are far away from the rest of the data and the local outliers are generally the ones located in areas of relatively low density. LOF is one such method that finds local outliers as well. It was shown by [9] that the global and local outliers are not strictly different by there are varying degrees of locality.

Other related works have generally focused on efficient implementation of the techniques discussed above, as in rather than improving the outlier detection method they make sure that the current methods are efficient.

Other methods such as ensemble techniques have a potential to improve outlier detection since, each technique captures different things/distributions about the data. Feature bagging is a common procedure which is used for clustering based ensemble methods. The subsampling based method [10] shows that ensemble methods can be applied in an efficient and effective way while without subsampling the data, ensemble methods might be costly in terms of the computation since every method has to be run separately.

Another interesting method was proposed by [6] where they show rather than using a single-view of the data, they use multiple views and although the problem becomes more complex due to the inconsistent behaviour of the samples across multiple views. They propose a multi view low rank analysis framework which contains robust data representation.

4 METHODS, ARCHITECTURE, AND DESIGN

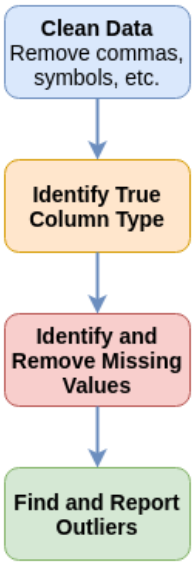
Our code can be found here. For reasons described below, we have created indexed data sets which is made available under:

```
/scratch/dam740/GROUP7_rid_gz/
```

The main focus in our entire methodology is on making all functions generalizable across columns, data sets, and data set collections. An overview of the methodology has been shown in the flowchart below; the specific ways of doing so are described in the following sections.

4.1 Data Cleaning and Null Value Removal

We have implemented several wrapper functions that automatically identify the true data types for a column. For this, we randomly sample 500 rows per column, apply regular expressions that remove commas (in the middle of a string) and all (non-alphanumeric) symbols like "$", "@", etc. from all them, and finally check if at least 70% of the points are compatible with a numeric type. Also, to make data handling easier by reducing the overhead amount of data stored and communicated at each step, we added a new column of indices (starting with 0) for all data sets. Please note, this is a necessity for running our code, which is why we have made the new indexed data available (see above for path). Any new data sets should also have the indices present in it before analysis. At present, we drop all null values in the data sets.



4.2 Outliers

For outlier detection, our analysis is divided into 2 main subgroups – numeric and strings.

- **IQR Rule (Numeric Columns)**
  For numeric columns, we use the approach used in box-plots or whisker-plots, where an outlier is is found as follows:
  Let Q_1 = First Quartile, Q_2 = Second Quartile, Q_3 = Third Quartile.
  And, $IQR = Q_3 - Q_1$
  Then, a given value V is an outlier if:
  (1) $V < Q_1 - 3 \times IQR$, or
  (2) $V > Q_3 + 3 \times IQR$

- **KMeans Clustering (Numeric Columns)**
  We perform KMeans clustering on all numeric columns and compute the euclidean distances of each point to its designated cluster center. To find outliers, we then find points that satisfy the IQR rule (in its own cluster) and report them.

- **Category Count (String Columns)**
  For strings, one possible approach as described in [3] is to compute the counts per category of a string class, and report the ones that satisfy condition (1) of the IQR rule. For instance, this may be useful for zip-codes, colors, days, etc. where some categories appear extremely rarely.

5 PRELIMINARY RESULTS

As mentioned earlier, the results are reproducible, as far as the indexed data sets are used as input. To obtain the results described below, the following commands may be used (back to back):

```
python3 clean_data.py bss9-579f.tsv
python3 get_outliers.py bss9-579f_clean.tsv
```

For obvious reasons, it is not possible to list out all results here. The reader may download the code from here and run the above

command to obtain the full list of outliers.

A short list of results follows.

For the cleaning data part, we show below an example of a column that is originally identified with a string data type. However, in reality it has an integer data type. The issue is that the values are separated by commas. From the figure below one can see that our method is able to detect this and correct it. Finally, our method removes all the null values, as seen below.

```
df.select('comparable_rental_3_full_market_value').head(11)
```

```
[Row(comparable_rental_3_full_market_value='2,405,000'),
 Row(comparable_rental_3_full_market_value='4,497,000'),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value='2,142,000'),
 Row(comparable_rental_3_full_market_value='9,226,000'),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value='9,226,000'),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value='5,357,000')]
```

```
replaced_df.select('comparable_rental_3_full_market_value').head(11)
```

```
[Row(comparable_rental_3_full_market_value=2405000.0),
 Row(comparable_rental_3_full_market_value=4497000.0),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=2142000.0),
 Row(comparable_rental_3_full_market_value=9226000.0),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=9226000.0),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=None),
 Row(comparable_rental_3_full_market_value=5357000.0)]
```

```
clean_df.select('comparable_rental_3_full_market_value').head(6)
```

```
[Row(comparable_rental_3_full_market_value=2405000.0),
 Row(comparable_rental_3_full_market_value=4497000.0),
 Row(comparable_rental_3_full_market_value=2142000.0),
 Row(comparable_rental_3_full_market_value=9226000.0),
 Row(comparable_rental_3_full_market_value=9226000.0),
 Row(comparable_rental_3_full_market_value=5357000.0)]
```

For numeric columns, we show the outliers for the column `num_level_3` from the `usap-qc7e_clean.tsv` data set:

summary	num_level_3
count	5302
mean	239.7766880422482
stddev	267.1447248550003
min	0.0
max	2020.0

rid	num_level_3
847	1386.0
851	1434.0
963	1337.0
990	1344.0
1493	1522.0

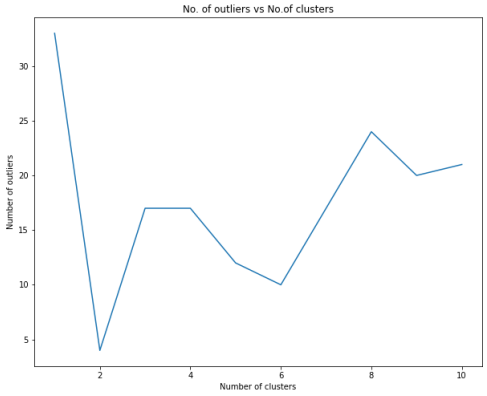
num level 3 counts: 38 counts % = 0.7167106752168994%

On the left, we show the summary statistics for the column, and on the right, the potential outliers for that column. As is evident, these examples fall very clearly out of the normal range of values, and hence, it's reasonable to say that they are outliers.

Next, we show the variation of the number of outliers detected with the number of clusters we set in the KMeans algorithm for the column `comparable_rental_2_gross_sqft` of the data set `bss9-579f.tsv`.

6 THINGS TO FOLLOW

In the next half of this study, we aim to extend this analysis to all data sets, i.e., increasing our definition of what comprises a null value or an outlier. And to employ more sophisticated methods like



Gaussian mixture models, Kullback Leibler divergence, and other multivariate methods for finding outliers.

REFERENCES

[1] Jason Brownlee. 2016. How To Handle Missing Values In Machine Learning Data With Weka. (Jun 2016). https://machinelearningmastery.com/how-to-handle-missing-values-in-machine-learning-data-with-weka/

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2007. Anomaly Detection: A Survey. (2007).

[3] Google. 2018. Locate Outliers, Google Cloud Dataprep Documentation, Google Cloud. (2018). https://cloud.google.com/dataprep/docs/html/Locate-Outliers_57344572

[4] Ming Hua and Jian Pei. 2007. Cleaning Disguised Missing Data: A Heuristic Approach. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, New York, NY, USA, 950–958. https://doi.org/10.1145/1281192.1281294

[5] Ming Hua and Jian Pei. 2008. DiMaC: a disguised missing data cleaning tool. In *KDD*.

[6] Sheng Li, Ming Shao, and Yun Fu. 2015. Multi-view low-rank analysis for outlier detection. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 748–756.

[7] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. ACM, New York, NY, USA, 427–438. https://doi.org/10.1145/342009.335437

[8] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.* 29, 2 (May 2000), 427–438. https://doi.org/10.1145/335191.335437

[9] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. 2014. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery* 28, 1 (01 Jan 2014), 190–237. https://doi.org/10.1007/s10618-012-0300-z

[10] Arthur Zimek, Matthew Gaudet, Ricardo J.G.B. Campello, and Jörg Sander. 2013. Subsampling for Efficient and Effective Unsupervised Outlier Detection Ensembles. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 428–436. https://doi.org/10.1145/2487575.2487676