Detection of Outlier and Null values in Big Datasets

Methods for detection among numerical values and strings

Yassine Kadiri
New York University
Center for Data Science
New York, New York
yk1859@nyu.edu

Ravi Teja Gadde
New York University
Center for Data Science
New York, New York
rtg267@nyu.edu

Victor de Murat
New York University
Center for Data Science
New York, New York
vdm245@nyu.edu

ABSTRACT

This paper proposes methods to detect outliers and null values in big datasets, more specifically in datasets involving numerical values or strings.

CCS CONCEPTS

• **Information systems**; • **Data management systems**; • **Information integration**; • **Data cleaning**;

KEYWORDS

Outliers, Null, Data Cleaning, Clustering, Numeric, Strings

ACM Reference Format:
Yassine Kadiri, Ravi Teja Gadde, and Victor de Murat. 2018. Detection of Outlier and Null values in Big Datasets: Methods for detection among numerical values and strings. In *Proceedings of Big Data (DS-GA 1004)*. ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Outlier detection is one of the most extensively studied topic in Data Mining. Outliers are usually classified as data instances or points that do not confirm with the expected behavior of the dataset. Detecting outliers and null values in datasets is a very important preprocessing step as they will lead to bad classification results, incorrect analysis of the data or sometimes give important information such as anamolous events in the data. Sources of outliers in datasets are mainly two types intrinsic behaviour of data or human errors while entering the data. In this project we are going to introduce different methods that we consider relevant to detect Null values and outliers. We restrict ourselves to finding outliers that belongs to numeric, string, or time stamp data types. For testing the effectiveness of our algorithm we manually injected outliers in to the datasets we have and checked if they have been classified as outliers along with manual inspection of the other outliers the algorithm has identified. We also evaluate the performance of our algorithms as we increase the size of the dataset.

2 PROBLEM FORMULATION

Outliers in datasets are classified in to three categories: point anomalies, contextual anomalies, collective anomalies. An individual data instance univariate or multivariate is called a point anomaly if it deviates from other instances in the dataset. Data instances that are anomalous with respect to specific context determined by their behavioral attributes such as time, location, value are termed as contextual anomalies. If a collection of data instances that are not independent of each other are anomalous they are termed as collective anamolies. Several supervised, semi-supervised and unsupervised learning techniques are proposed for outlier detection. Supervised learning techniques involve classifying instances in to normal or anamolies using classification algorithms such as multilayer perceptrons, support vector machines, bayesian networks etc. Unsupervised learning techniques rely of distance based methods such as k nearest neighbours and clustering to detect outliers. kNN based techinques use thresholding on the distance to the kth nearest neighbour to detect anamolies with an assumption that distance to the nearest neighbours would be far for anamolies and vice versa for normal data instances. Clustering based techniques classify instances that do not belong to any cluster, instances that are far from cluster centroid, or instances that belong to sparse clusters as outliers. Parametric statistical learning techniques fit a probability distribution to the data and classify instances with low probability as outliers.

In this paper, we try to detect collective and point anomalies for data types that belong to numerical, string or timestamp category. We use unsupervised learning techniques such as Gaussian Mixture Models, k-Nearest Neighbors, Histograms, K-means to detect outliers in the numerical data. For string values, we distinguish two cases: short strings, which suggest that those strings represent categories, and long strings. We use Frequency based approaches and distance based algorithms using Tf-Idf to detect outliers in strings. For dates we resolve to simple rule based approach based on frequency of instances for identifying null values and outliers.

3 RELATED WORK AND REFERENCES

Lot of work has been done on outlier detection. [1] has done an extensive survey on outlier detection highlighting different kinds of outliers, different types of input data and different techniques used for this problem. [2] has proposed pruning techniques for efficiently detecting outliers in large datasets using nearest neighbor techniques. Several authors worked on classification based techniques for detecting outliers. [3, 4] used one class SVMs for detecting outliers, [5] used multi-layered perceptrons for detecting outliers. Distance based techniques such as k nearest neighbors

[6, 7] and clustering [7, 8] are widely popular for outlier detection. Several authors also proposed parametric statistical learning techniques [9] to detect outliers by fitting probability distributions to data. Lot of work has been done on different applications of outlier detection such as fraud detection, intrusion detection etc. [10–12]. Other works we studied model the normal instances as a mixture of parametric distributions. The main idea of those works is that a test instance which does not belong to any of the learnt models is declared to be anomaly. Gaussian mixture models have been mostly used for such techniques [14], and have been used to detect strains in airframe data [15, 16], to detect network intrusions [17, 18] and also to detect anomalies in mammographic image analysis [19, 20]. Such works therefore confort us in the idea that trying to fit a distribution to the data in order to detect outliers makes sense and actually performs well.

4 METHODS, ARCHITECTURE AND DESIGN

In order to detect outliers in the dataset we iterate though each column and randomly sample few elements from it. We use regular expressions to detect if the columns are either string, numerical or time stamp based on these samples by identifying the most frequent type among the samples. If the most frequent type exceeds our threshold we will classify the column as belonging to the data type. We use a threshold of 0.8 for this purpose. We have worked mainly on three data-types: numerical, string and timestamp data. Once we identified the data type we detect the Null values present in the dataset. We then filter out Null values and pass this column to our outlier detection algorithms based on the data type. We have used Apache Spark MLlib for our algorithms.

4.1 Null Value detection

We detect null values in numeric and timestamp data using the following approach. We assume that numeric and timestamp data usually encompass a wide range of values resulting in very less frequency for each distinct value present in the column. If an outlier is present in the column it is expected to have higher frequency than the other real values present in the column. If this frequency is too large compared to the second most frequent element in the column we detect this as Null value or an outlier. However, our method will not be able to detect null values if the column contains too few null values or entries in the column have very high repetitions among values. We have observed that such cases are very less in the real world datasets. We also filter out values that are not Numerical or Timestamp based on regular expressions. For strings we filter out values that are either empty or with content 'NA'.

4.2 Methods for Numerical Data

For numerical data, we have three types of outliers which are shown in the following figures. Our methods aim to spot those types of outliers.

4.2.1 Point anomalies. The first method we implemented was simply to fit a Gaussian to our data set. Our goal is to detect outliers of the type present in Figure 1 a. The approach here is to look for points whose distance to the mean μ exceeds $\delta\sigma$ where δ is a distance factor that we choose. This distance factor is actually such that for $\delta = 2$ we exclude 95.5% of the data and 99.7% for $\delta = 3$. We use $\delta = 4$ for as our threshold. The idea here is to exclude values that are at the margins of the distribution and that we can safely call outliers. We did this using the GaussianMixtureModel from spark.mllib library and by setting the number of Gaussians in this model to 1. With this method we can detect outliers that take extreme values and proceed further to detect outliers of other types.

After applying the first filtering method, inside the dataset we might still have point anomalies as shown in the Figure 1 b. In order to detect such outliers we used a k-Nearest Neighbors approach on the 'filtered' dataset to detect points that are far enough from its neighbors so that we can classify them as outliers. Indeed, our second method consists of computing for each point the mean distance with its k nearest neighbors (with L1 distance as it is more relevant in 1 dimension) and drawing a distribution with these. As we expect these mean distances to follow a distribution closely related to a Gaussian, the criteria we chose is based on its mean and standard deviation. For a Gaussian distribution we have : $P(\mu - 3\sigma < X < \mu + 3\sigma) = 0.9973$. Hence, we detect an outlier when the population of some category is inferior to $\mu - 3\sigma$. After some tests, we finally set $k = 20$ for this method. However, this method is not efficient in a situation where the outliers are numerous and close together as shown in Figure 1 c. That is why we built a method on top of these two to detect these collective anomalies type of situations.

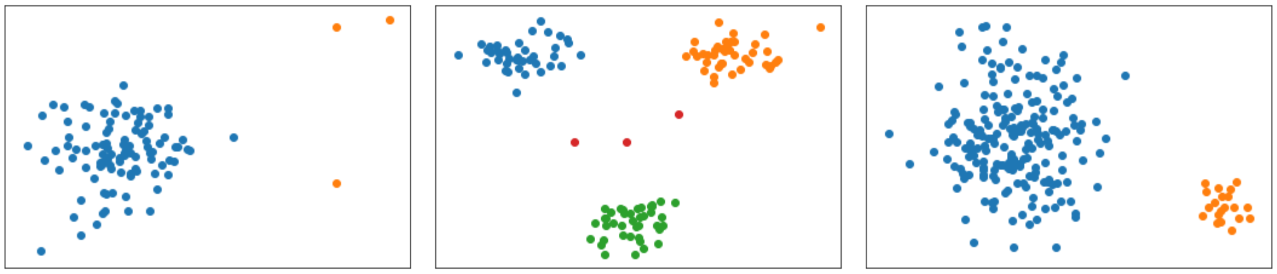
4.2.2 Collective anomalies. For collective anomalies, we designed two different methods that involve clustering our data in order to spot the clusters of outliers. The first one is Gaussian Mixture Model, and the second one is k-Means. This approach detects outliers of type shown in the Figure 1 c.

The Gaussian Mixture Model approach is quite natural. The idea in this approach is to fit k Gaussian to our data. The main assumption here is that with a correctly chosen k, we expect to separate the outliers in distinct clusters. The main strength of this method as opposed to other clustering methods is that with Gaussians, we will be able to compute the likelihood for each point once we know to which of the k Gaussians each point belongs. In our first try, we attempted to produce a 'k-Gaussian clustering'. Once fitted, we have access to the weight of each of the k Gaussians, the means, standard deviations of the Gaussians and also the labels of each point in our data set, i.e. to which of the k Gaussians a point belongs. We call $w^{(k)} = (w_1^{(k)}, w_2^{(k)}, ..., w_k^{(k)})$ the vector of weights, $L^{(k)} = (l_1^{(k)}, l_2^{(k)}, ..., l_k^{(k)})$ the vector of labels, $\mu^{(k)} = (\mu_1^{(k)}, \mu_2^{(k)}, ..., \mu_k^{(k)})$ the vector of means for the k Gaussians and $(\sigma^{(k)} = \sigma_1^{(k)}, \sigma_2^{(k)}, ..., \sigma_k^{(k)})$ the vector containing the standard deviations. First, when fitted, we define a parameter $p \in [0, 1]$ such as, for any i if we have:

$$w_i^{(k)} < \frac{p}{k} \tag{1}$$

we reject the cluster i and consider all its values as outliers. In our tests, we generally took $k = 5$ and $p = 0.1$ which means we reject if for any i we have $w_i^{(k)} < 0.02$.

Once this is done, we take the remaining m Gaussians and their corresponding vectors of means $\mu^{(m)} = (\mu_1^{(m)}, \mu_2^{(m)}, ..., \mu_m^{(m)})$. The



a) Point anomalies far from data **b) Point anomalies in between data** **c) Collective anomalies**

Figure 1: The different types of outlier situations for numerical data. The first two were addressed with our point anomalies detection methods. The last one correspond to an collective anomalies situation.

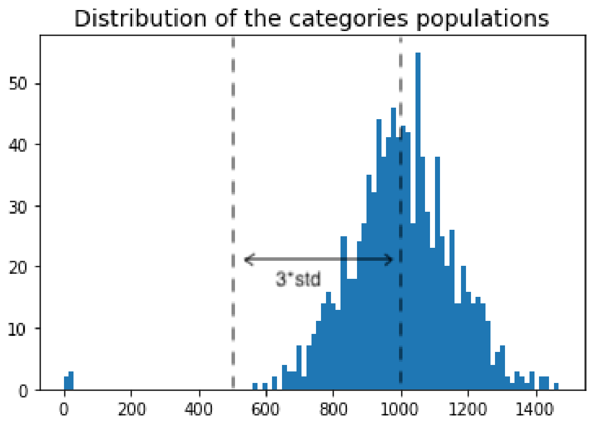


Figure 2: The distribution of the populations is closely related to a Gaussian. Outliers are detected here.

idea here is to sort the vector of means and make sure that we don't have any extreme values remaining in our dataset. To do so, we'll name $\mu_\pi^{(m)} = (\mu_{\pi(1)}^{(m)}, \mu_{\pi(2)}^{(m)}, ..., \mu_{\pi(m)}^{(m)})$ the sorted vector of means (π is the permutation that sorts the values) and we will perform some comparisons. We therefore came up with the following comparisons, introducing *mag* a magnitude constant in order to reject or accept the extreme value. We also translated the values of $\mu_\pi^{(m)}$ so that they're all positive and $\mu_\pi^{(m)}(1) = 1$. So if:

$$mag \times \frac{\mu_{\sigma(m-1)}^{(m)}}{\mu_{\sigma(1)}^{(m)}} < \frac{\mu_{\sigma(m)}^{(m)}}{\mu_{\sigma(m-1)}^{(m)}} \tag{2}$$

then it means that the largest value of $\mu_\sigma^{(m)}$ is too large meaning that the corresponding Gaussian contains outliers. But if this is not observed and on the other hand we have:

$$\frac{\mu_{\sigma(2)}^{(m)}}{\mu_{\sigma(1)}^{(m)}} > mag \times \frac{\mu_{\sigma(m)}^{(m)}}{\mu_{\sigma(2)}^{(m)}} \tag{3}$$

then it means that the smallest value of $\mu_\sigma^{(m)}$ is too small, meaning that the corresponding Gaussian contains outliers. To sum up, this method we devlopped requires 3 hyperparameters: k, p and *mag*

Another method we developped for collective outliers involves using k-Means. The idea with this algorithm is similar to the previous one and it is that with an appropriately chosen hyperparameter k, fitting the k clusters to the data will result in identifying the clusters that are far from other clusters. Outliers that are repeated multiple times in data and do not fit the general data distribution will be clustered far from other clusters. We implemented this approach with k = 2 and identified one of the clusters as outliers if that cluster has very few instances compared to the rest of the data. If the proportion of one cluster is smaller than say $t = 0.02$, then we consider this cluster to be the cluster of outliers while the other one contains the inliers. This threshold was chosen arbitrarily as it is difficult to choose a value that would be optimal especially as no criterion exists to set such parameters. Also, this is based on our understanding and modeling of how numerical outliers generally behave but there might be other cases for which such an approach would need to be re ned. In such a situation, the idea would then be to alter the hyperparameter k as this would help refine the outliersâĂŹ detection. Obviously, we will set a limit to how much k can grow in order to avoid overfitting.

4.3 Methods for Strings

For text data, we concluded that data could be split in two different categories: short textual description (less than 3 words) and real texts (reviews, etc.). Indeed, depending on the result of this split, detection methods differ dramatically. Columns with short textual data will often refer to categorical features whereas columns furnished with long texts will be very likely to be reviews or descriptions. Hence, outliers for these two types of cases are defined in a different way : for short texts, they correspond to categories with abnormal names and can be detected by the fact that they should have very low population. On the other hand, for long text, outliers will consist in weird sentences not related to most of the other texts in the column. In the first case, we decided to look at the number of distinct elements and treat these as categories. Intuitively, as explained above, we defined a category whose population is very small as an outlier. To be more rigorous, we designed the following algorithm: we first identify all the categories. Then, we compute the distribution of the populations of these categories and we look for the 15% least populated categories. Among these categories, we merge the ones that have small edit distance with some other category (cleaning procedure - to find in these 15% the ones that are really outliers, not due to spelling mistakes) and we report the categories we couldn't merge as outliers. Then, for long text data (more than 3 words), we used the MLLIB TF-IDF featurizer. We then applied an l2 k-nearest neighbor approach in this feature space. For each point, we computed the average distance of it with its k nearest neighbors and determined that it corresponds to an outlier if this distance is over the average of the distances over all the points plus two times the standard deviation. This makes sense as we expect the distribution of the average k-nearest distances to be close to an exponential distribution. The main limitations around this method are the dimension of the resulting embeddings and the difficulty to set an exact threshold to discriminate outliers.

Let's discuss the k-Means approach first. The idea with this algorithm is that with an appropriately chosen hyperparameter k, fitting the k clusters to the data will result in identifying the clusters that are far from other clusters. Outliers that are repeated multiple times in data and do not fit the general data distribution will be clustered far from other clusters. We implemented this approach with $k = 2$ and identify one of the clusters as outliers the distance between them is far and one of them have very less values in it. If the proportion of one cluster is smaller say 0.05, then we consider this cluster to be the cluster of outliers while the other one contains the inliers. This threshold was chosen arbitrarily as it is difficult to choose a value that would be optimal especially as no criterion exists to set such parameters. Also, this is based on our understanding and modeling of how numerical outliers generally behave but there might be other cases for which such an approach would need to be refined. In such a situation, the idea would then be to alter the hyperparameter k as this would help refine the outliers' detection. Obviously, we will set a limit to how much k can grow in order to avoid overfitting. In the future, the threshold t will be refined as it will be tested on all the datasets available and compared with other techniques.

5 RESULTS

Inorder to test the outliers we manually injected outliers in the columns of the database to see if the algorithm are working. We have also analyzed the outliers identified by the algorithm which are present in the original data. For the text data we have observed that our algorithms actually retrieve the outliers we injected. For numerical data too we have followed the same approach and achieved decent results after manual inspection. The results are formulated in Table 1. We have also observed the time complexity of our algorithm as we increase the size of the dataset. Inorder to avoid bias we have duplicated the same dataset n number of times and reported the time complexity as a function of n in Table 4. We have ignored performance for kNN approach and for long strings as the number of datasets containing outliers of that type are very limited. The number of rows are shown in the table. The number of tasks are automatically decided by Spark.

Table 1: Outliers Identified

Injected	Detected	Other Outliers	Mean,Std of Data
1000.0	1000.0	320.0, 110.0	3.67, 15.86
-100	-100	515.0, 516.0	34.9, 55.6
20140000.0 (1176 times)	20140000.0	-	-
Abracadabra	Abracadabra	-	-
2078-09-02T06:45:00	2078	2020	-

Table 2: Time Analysis Numeric

num rows	n	time 1 Gaussian	time k Gaussians
23749	1	39.655s	52.133s
23749*2	2	45.395s	76.663s
23749*4	4	90.697s	120.997s

Table 3: Time Analysis String

num rows	n	time in seconds
55095	1	35
55095*2	2	40
55095*4	4	52

Table 4: Time Analysis Date

num rows	n	time in seconds
87755	1	53
87755*2	2	73
87755*4	4	111

6 CODE AND APPROACH

We have developed our models using Spark. We have converted the columns of the database in to RDDs and iterate through them to identify outliers in each column. We have used MLLib library of spark to run most of our algortihms. Our code base can be found here https://github.com/yasskad/Big-Data

7 CONCLUSIONS

We have experimented with a few techniques that are relevant for finding outliers. We are able to verify that our algorithms were indeed able to detect the outliers present in the dataset. Our work can be extended to find better ways to detect collective anomalies as the methods we tried have a bit high time complexity. Our methods for string and point anamolies in numeric scales with data.

REFERENCES

[1] Chandola, V., Banerjee, A. and Kumar, V., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), p.15.

[2] Ramaswamy, S., Rastogi, R., and Shim, K. (2000, May). Efficient algorithms for mining outliers from large data sets. In ACM Sigmod Record (Vol. 29, No. 2, pp. 427-438). ACM. Chicago

[3] Wang, Y., Wong, J., and Miner, A. (2004, June). Anomaly intrusion detection using one class SVM. In Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC (pp. 358-364). IEEE.

[4] Davy, M., and Godsill, S. (2002, May). Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation. In Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on (Vol. 2, pp. II-1313). IEEE.

[5] Barson, P., Field, S., Davey, N., McAskie, G., and Frank, R. (1996). The detection of fraud in mobile phone networks. Neural Network World, 6(4), 477-484.
[6] Byers, S., and Raftery, A. E. (1998). Nearest-neighbor clutter removal for estimating features in spatial point processes. Journal of the American Statistical Association, 93(442), 577-584.
[7] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In Applications of data mining in computer security (pp. 77-101). Springer, Boston, MA.
[8] Brockett, P. L., Xia, X., and Derrig, R. A. (1998). Using Kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. Journal of Risk and Insurance, 245-274.
[9] Laurikkala, J., Juhola, M., Kentala, E., Lavrac, N., Miksch, S., and Kavsek, B. (2000, August). Informal identification of outliers in medical data. In Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology (Vol. 1, pp. 20-24).
[10] Ghosh, S., and Reilly, D. L. (1994, January). Credit card fraud detection with a neural-network. In System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on (Vol. 3, pp. 621-630). IEEE.
[11] Wong, W. K., Moore, A. W., Cooper, G. F., and Wagner, M. M. (2003). Bayesian network anomaly pattern detection for disease outbreaks. In Proceedings of the 20th International Conference on Machine Learning (ICML-03) (pp. 808-815).
[12] Fawcett, T., and Provost, F. (1999, August). Activity monitoring: Noticing interesting changes in behavior. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 53-62). ACM.
[13] Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sander, J. (2000, May). LOF: identifying density-based local outliers. In ACM sigmod record (Vol. 29, No. 2, pp. 93-104). ACM.
[14] Agarwal, D. 2006. Detecting anomalies in cross-classified streams: a bayesian approach. Knowl- edge and Information Systems 11, 1, 29âĂŞ44.
[15] Hickinbotham, S. J. and Austin, J. 2000. Novelty detection in airframe strain data. In Proceedings of 15th International Conference on Pattern Recognition. Vol. 2. 536âĂŞ539.
[16] Hollier, G. and Austin, J. 2002. Novelty detection for strain-gauge degradation using maxi- mally correlated components. In Proceedings of the European Symposium on Artificial Neural Networks. 257âĂŞ262âĂŞ539.
[17] Yamanishi, K. and ichi Takeuchi, J. 2001. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, 389âĂŞ394.
[18] Yamanishi, K., Takeuchi, J.-I., Williams, G., and Milne, P. 2004. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Mining and Knowledge Discovery 8, 275âĂŞ300.
[19] Spence, C., Parra, L., and Sajda, P. 2001. Detection, synthesis and compression in mammo- graphic image analysis with a hierarchical image probability model. In Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. IEEE Computer Society, Washington, DC, USA, 3.
[20] Tarassenko, L. 1995. Novelty detection for the identification of masses in mammograms. In Proceedings of the 4th IEEE International Conference on Artificial Neural Networks. Vol. 4. Cambridge, UK, 442âĂŞ447.