# Big Data Final Project

# Implementing Local Outlier Detection (LOF) on Hadoop or Spark

### Team 1 Emily Hao, Daojun Liang

### DS503 Big Data Management

### 12/10/2019

## Table of Content

[Table of Content](#_qt7tfe2xsycm) 2

[Introduction](#_ilt2yghoqelm) 3

[Related Work](#_ssx472gqzj93) 3

[2.1 LOF Identifying Density-Based Local Outliers](#_vwctffdu49et) 4

[2.2 Algorithms for Mining Distance-Based Outliers in Large Datasets](#_3t6qkr4g9krh) 5

[2.3 Distributed Local Outlier Detection in Big Data](#_42mp99flcgvu) 7

[Preliminaries [definitions]](#_1lmdefvzgwuc) 7

[Challenges](#_p9mp8sr8e5oy) 8

[Implementation](#_29ej9yd01l4g) 9

[Centralized LOF](#_wujtdimcamtz) 9

[Data Driven KNN Search](#_10a7e54hchs) 10

[Data-Driven Distributed LOF](#_v4o3y622jg3) 11

[Performance Evaluation](#_ouojx296gf6i) 13

[Reference](#_2c31dzj9jed1) 16

## Introduction

In data mining, anomaly detection, also called outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data[1]. This project, the team focuses on the local outlier factor (LOF), which is an algorithm proposed by Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander in 2000. The purpose of the creation of LOF algorithm is for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours[2].

The basic idea of LOF is that it is detecting outliers based on a concept of a local density, and we use the distance of k nearest neighbors to estimate the density of the locality. We can compare the local density of an object to the local densities of its neighbors to identify regions of similar density. Furthermore, we can detect the points that have a substantially lower density than their neighbors, and those are considered to be outliers. In this project, we are defining the local density as the estimated typical distance at which a point can be "reached" from its neighbors. We also define the term reachability distance as an additional measure to produce more stable results within clusters.

## Related Work

For background research for this project, we researched and studied through density-based LOF, distance-based LOF, and distributed LOF. In this section, we will introduce the concept and definitions of density-based LOF and the differences between density-based LOF and distance-based LOF. We would also introduce the concept and definitions of distributed LOF.

### 2.1 LOF Identifying Density-Based Local Outliers

According to “LOF: Identifying Density-Based Local Outliers”, the definition of outliers are objects not located in clusters of a dataset, usually called noise. The Hawkins-Outlier definition is stated that “an outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism”.[2] The difference between the definitions of outliers is that the definition of outlier proposed in this paper is that being outlying is not a binary property. Instead, this paper proposed to assign each object an outlier factor, which is the degree the object is being outlying. The basic idea proposed in this report of finding outliers is by introducing a local outlier for each object in the dataset, indicating its degrees of outlierness. The outlier factor is local in the sense that only a restricted neighborhood of each object is taken into account. The LOF of an object is based on the single parameter of MinPts, which is the number of nearest neighbors used in defining the local neighborhood of the object.

The paper showed that for most objects p in a cluster, the LOF of p is approximately equal to 1. And for other objects, the general upper and lower bound are tight for important classes of objects. For other classes of objects, another theorem was proposed for specifying better bounds.

Theorem 1: Let p be an object from the database D, and . The result derived by the proof is . It is a function of the reachability distance in p’s direct neighborhood relative to those in p’s indirect neighborhood.

In general, if an object p is not located deep inside a cluster, but its MinPts-nearest neighbors all belong to the same cluster, the proposed theorem estimated the LOF very well, as the minimum and maximum LOF bounds are close to each other. For other objects, the paper proposed theorem 2 to establish two sets of upper and lower bounds on the LOF, depending on whether the MinPts-nearest neighbors come from one or more clusters. Next, we need to determine the right MinPts values. This paper proposed to determine a range of MinPts values and compute for each object its LOF values within the range. [4] Further, we can determine the ranking of an object p based on , where MinPtsLB and MinPtsUB are the upper and lower bounds of the MinPts range. The paper proposed to take the maximum to highlight the instance at which the object is the most outlying. The experimental results demonstrate the approach of finding local outliers is efficient for datasets where the nearest neighbor queries are supported by index structures and practical for large dataset.

### 2.2 Algorithms for Mining Distance-Based Outliers in Large Datasets

In summary, the definition of finding distance-based outliers is through comparing similarity between two objects by measuring the distance between the two objects in data space, if this distance exceeds a particular threshold, then the data object will be called as the outlier.[5] Based on the report “Algorithms for Mining Distance-Based Outliers in Large Datasets”, Knorr and Ng proposed the notion of distance-based outliers in 1998. There are many algorithms under distance-based LOF. One of the most popular and simple to implement is K neighbor technique. This technique operates under the assumption that normal points have several closely located neighbors, while outliers are located far from other points. In this paper, Knorr and Ng experimented with two simple O(K N^2) algorithms. They also developed a cell-based algorithms that are linear with respect to N and are suitable for K4. The cell-based algorithm developed for large, disk-resident datasets also guarantees that no data page is read more than 3 times, if not once or twice.

The differences between density-based LOF and distance-based LOF are summarized below. Distance-based LOF notion presented in the paper “Algorithms for Mining Distance-Based Outliers in Large Datasets” generalizes many notions from the distribution-based approaches and enjoys better computational complexity than the depth-based approaches for larger values of k. The notion of distance based outliers is extended by using the distance to the k-nearest neighbor to rank the outliers. Although the distance-based LOF technique is simple to implement and efficient to a considerable approach, it does not account for the “locality” of the data element which can lead to misjudging the outlier in certain kinds of datasets. Furthermore, in general, distance-based techniques are typically expensive and hence are not applied in scenarios where computational complexity is an important issue[3].

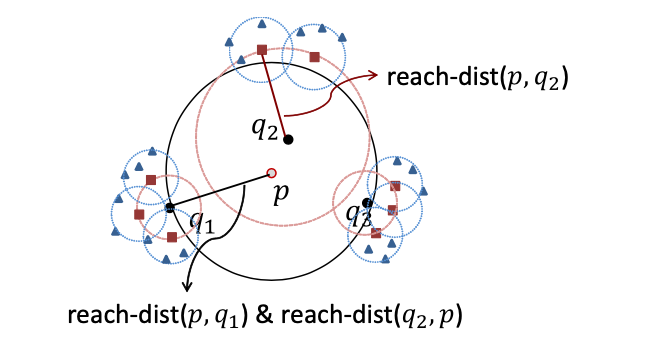
### 2.3 Distributed Local Outlier Detection in Big Data

The paper, "Distributed Local Outlier Detection in Big Data", proposed DLOF. The simple definition of DLOF is that in the LOF computation process of point p, although each step requires different types of intermediate values, these values are always associated with a fixed set of data points. Therefore, utilizing that observation, we can ensure the input data and required intermediate values are co-located on the same machine in the computation pipeline[6].

## Preliminaries [definitions]

We summarized the notions used in this report into the table below for reference, based on the assumption that o, p, q are used to denote objects in a dataset and c is used for a set of objects.

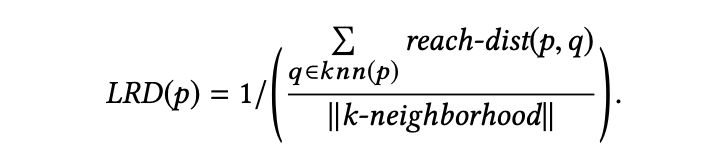
|  |  |
| --- | --- |
| Denotations | Definitions |
| d(p,q) | Distance between objects p and q |
| d(p, C) | The minimum distance between p and  object q in C |

We also listed the core definitions we used for our implementations. 

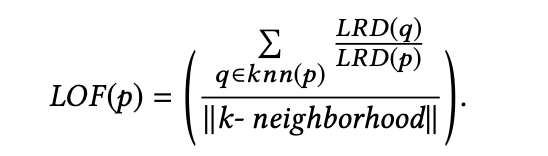
*Definition 3.1.* The k-distance of a point is the distance between p and a point such that for at least k points , and for at most k-1 points , .

*Definition 3.2.* Given points where , the Reachability Distance of p w.r.t. q is defined as:

*Definition 3.3.* Given points , where , the Local Reachability Density (LRD) of p is defined as:



*Definition* 3.4. Given points , where , the Local Outlier Factor (LOF) of p is defined as:



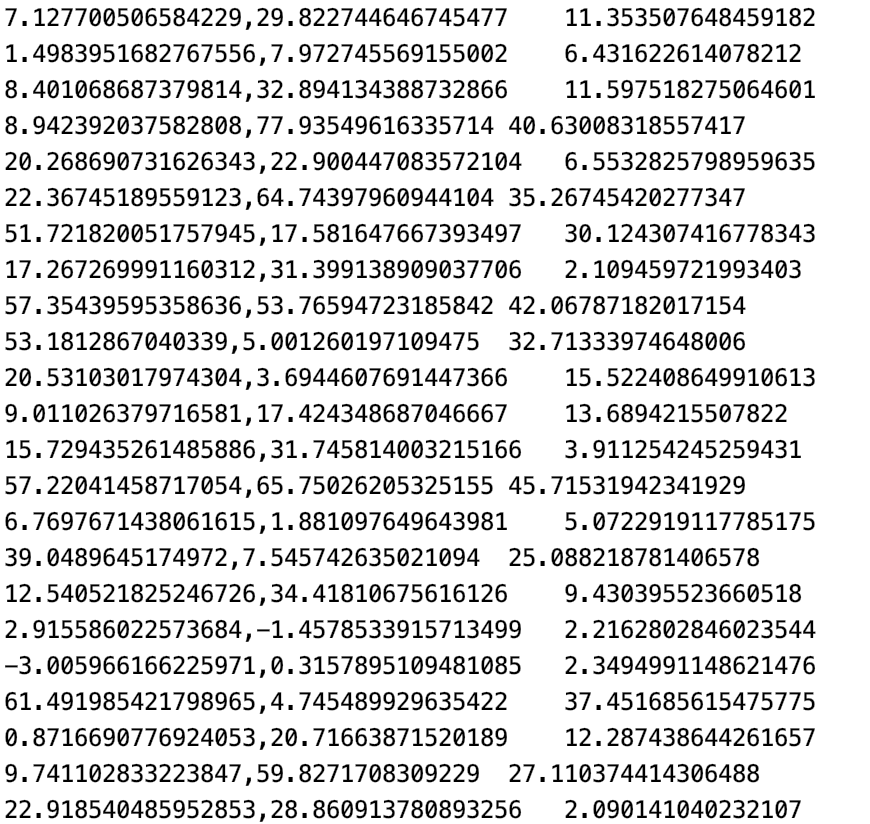
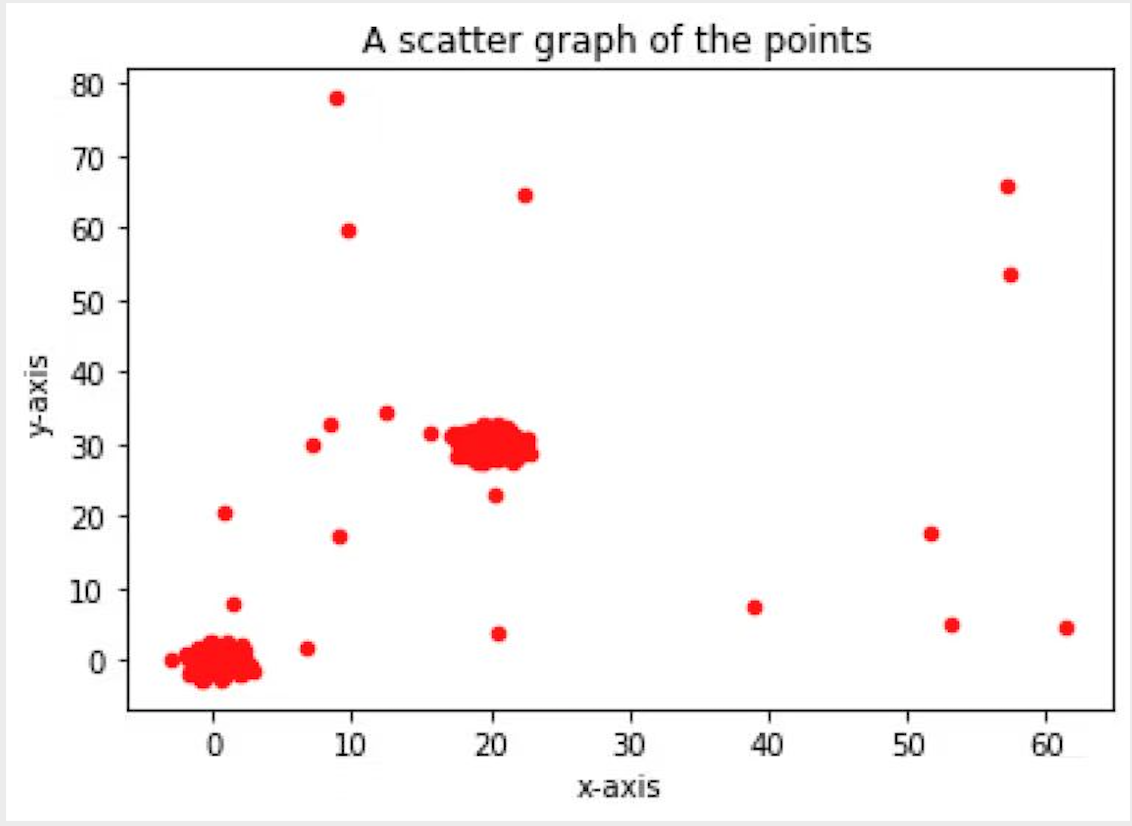
## Challenges

For this project, the team aimed to build a good understanding on the LOF algorithm and how it works in a centralized environment. We build and implemented a centralized LOF algorithm referencing the report "*LOF Identifying Density-Based Local Outliers*". The team also built a good understanding of distance-based LOF to further understand the pros and cons of LOF, referencing the report "*Algorithms for Mining Distance-Based Outliers in Large Datasets*". The team came up with a design for making LOF a distributed algorithm that can run on Hadoop with referencing the report "*Distributed Local Outlier Detection in Big Data*". The team also implemented early termination optimization to speed up the execution.

## Implementation

### Centralized LOF

We implemented centralized LOF on Hadoop with Map-Reduce Job. The centralized LOF implementation only takes one Map-Reduce Job. The mapper takes all data points as the input, and output <”1”,[list of all data points]>, which is passed to the reducer. The reducer would iterate through all the data points. For each points, the following computation is done. First, to compute the K-distance of the point p, we need to acquire the kth nearest neighbor(kNN) of point p. We stores p’s kNNs and K-distance for next computation. Next, we compute the LRD of point p by acquiring the K-distances of each of p’s kNNs. Lastly, we compute the LOF of point p by acquiring the LRD values and K-distances of each of p’s kNNs. In our reducer, we set the bound of LOF value to be 2, which means the points with LOF values greater than 2 would be labeled and outputted as outliers. Therefore, our reducer would output< outlier’s coordinate, LOF value>.



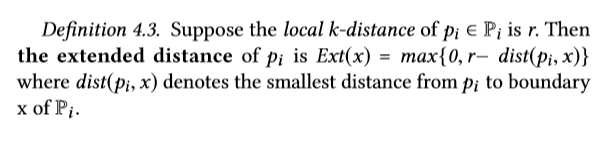
The left of Figure 2 shows the input data we passed in. We generated 2 clusters and 20 outliers. The right of Figure 2 shows the output from reducer. The centralized LOF we implemented detached and outputted 22 outliers, with most of the coordinates matches up with the inputted outliers.

### Data Driven KNN Search

Previous Distributed LOF use the furthest possible kNN of all points in a partition to bound to support points. This corresponds to a safe but worst case estimation, leading to a large number of replicas and therefore results in a high duplication rate. We implemented Data Driven kNN Search as the optimization to speed up our computation process. The benefit of implementing Data Driven kNN Search[7] is to reduce the size of the support areas. With reduced-size support areas, the algorithm can save time because this optimization reduced the duplicated rate of data, therefore, reduce the overall computation cost.

The core idea of data driven distributed LOF is that the “actual k-distance” of discovered in the whole dataset D cannot be larger than its local k-distance. (Here, local k-distance means that we only use the points in the same grid to calculate a point’s k-distance.) From this point of view, we only need to calculate a point’s extended distances[7] to find the support bounds.

The extended distance is defined as:



Extended distances describe how much further the boundaries of the original partition have to be expanded to form the support boundaries of the partition. For each grid in one partition, we can calculate its extended distances and then we calculate the maximum extended distances of all the points’ extended distances as the grid’s extended distances. In this way, we can calculate a grid’s support boundaries.

### Data-Driven Distributed LOF

The implementation of Data Driven Distributed LOF is done with four Map-Reduce Jobs and a preparation Map-Reduce Job. For the pre-processing part, we partitioned the datasets into 100 100x100 grids with the total area of 1000x1000. In our partition, we ensure that each grid has a distributed cluster. In the preparation phase, we labeled every grid with from 1 to 100, called CoreID. We also calculated support area of each grid and saved the data into a txt file called SupportBond.txt. This file would be passed into the mapper of the first Map-Reduce job as one of the inputs. Lastly, in the preparation phase, we find a list of each point’s support IDs.

First job is computing K-distance with the Core Partition KNN search. The purpose of this computation is to find the local kNNs and local k-distance of the data points. The mapper takes in all data points as the input, and output all the data points with their matching coreID (<coreID, points>). Reducer takes in Mapper’s output, and output <Nullwritable, “[Point, coreID, status, Local kNN, Local k-distance, Nminpts]”>.

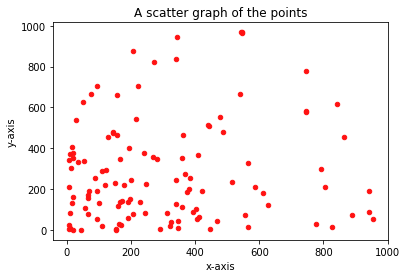
Second job is computing K-distance with Support Partition KNN search. The purpose of this Map-Reduce job is to update each point to the actual kNNs and actual k-Distance. The mapper takes in the output of last Map-Reduce job as the input. The mapper outputs either <coreID, “c:[Point, coreID, status, Local kNN, Local k-distance, Nminpts]”>, or <supportID, “s:[Point, coreID, status, Local kNN, Local k-distance, Nminpts]”>. The reducer would take those are inputs, and output <nullWritable, “[Point, Neighborhood, Actual K-distance, Nminpts]”>.

Third job is to compute LRD. The mapper takes in the output for second Map-Reduce job. The mapper output<>. The reducer takes those as the input, and output<NullWritable, “[Point, Neighborhood, LRD value, Nminpts]”>.

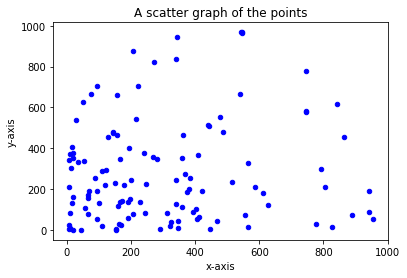
Fourth job is to compute LOF. The mapper takes in the output for third Map-Reduce job. The mapper output<>. The reducer takes those as the input, and output<outlier coordinates, LOF>.

## Performance Evaluation

In this section, we show that our implementation can be used in successfully identify outliers which appear to be meaningful. We implemented a 2-dimensional dataset. We used 100 gaussian clusters, each gaussian cluster has 50 data points, we have 100 outliers. The result of the centralized LOF is shown below.



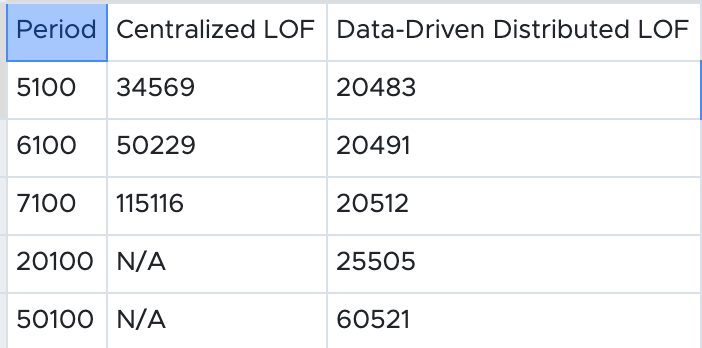
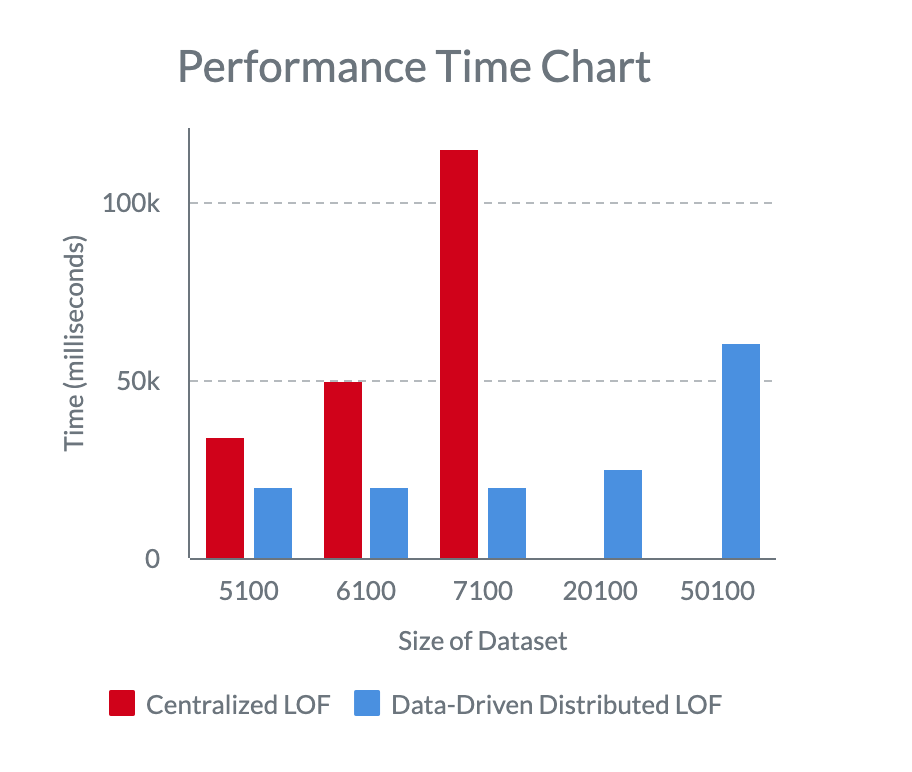
The result of the Data Driven Distributed LOF is shown below.



As we can see from the figures above, the results outputted by both the implemented LOF algorithm are the same. Therefore, both the implemented Centralized and Distributed LOF algorithm work efficiently.

We tested different sizes of datasets and the performance time for both Centralized and Distributed LOF as shown below.

This graph shows that Distributed takes much less time to run. In fact, the Centralized LOF took more than 10,000 millisecs to run for dataset of 7100 data points, but Distributed only took 20512.



In conclusion, our implementation of Centralized and Data-Driven Distributed LOF both work efficiently to detect outliers. However, Data-Driven Distributed LOF perform much faster than Centralized Distributed LOF on the same size of datasets.

## Reference

[1]. Zimek, Arthur; Schubert, Erich (2017), "Outlier Detection", Encyclopedia of Database Systems, Springer New York, pp. 1–5, doi:10.1007/978-1-4899-7993-3\_80719-1, ISBN 9781489979933

[2]. Breunig, Markus & Kriegel, Hans-Peter & Ng, Raymond & Sander, Joerg. (2000). LOF: Identifying Density-Based Local Outliers.. ACM Sigmod Record. 29. 93-104. 10.1145/342009.335388.

[3]. Mandhare, Harshada & Idate, S.. (2017). A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques. 931-935. 10.1109/ICCONS.2017.8250601.

[4]. Chen, L. (2019, November). Density-Based Outlier Detection. Lecture Notes. Hong Kong.

[5]. Ramaswamy, Sridhar & Rastogi, Rajeev & Shim, Kyuseok. (2000). Efficient Algorithms for Mining Outliers from Large Data Sets.. ACM SIGMOD Record. 29. 427-438. 10.1145/335191.335437.

[6]. Yan, Yizhou & Cao, Lei & Rundensteiner, Elke. (2017). Distributed Top-N local outlier detection in big data. 827-836. 10.1109/BigData.2017.8257998.

[7]. Yan, Yizhou & Cao, Lei & Kuhlman, Caitlin & Rundensteiner, Elke. (2017). Distributed Local Outlier Detection in Big Data. 1225-1234. 10.1145/3097983.3098179.