| | | |
|---|---|---|
| LAM Cheuk Fung | 20268763 | cflamad |
| SHUM Ka Hei | 20273251 | RonaldShum |
| NG Man Yan | 20277922 | cindyngng |
| LEE Ho Ting | 20278419 | htleeab |

# K-anonymity in distributed cluster

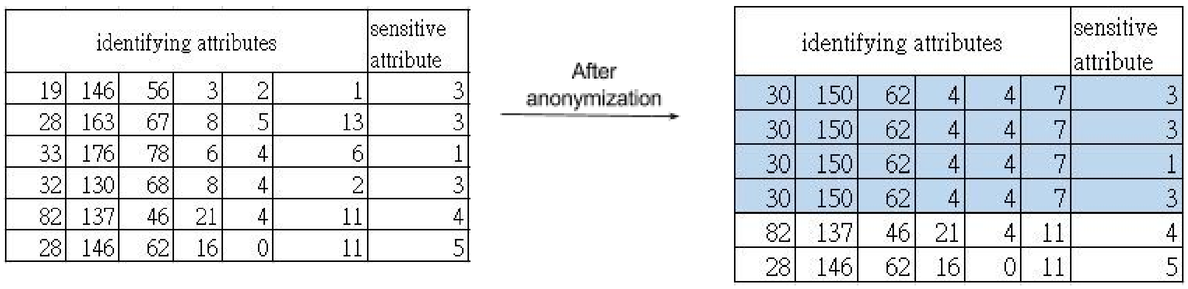
## Introduction

A huge amount of personal information is being collected every day. For example, in the medical field, personal health information collected by hospitals can help identify relationships between physical conditions and diseases. However, such disclosures of personal data raise serious privacy concerns. To alleviate such concerns, data owners should anonymize datasets before releasing them to the public. One popular anonymization approach is k-anonymity. A dataset is k-anonymized if at least k records are returned for all combination of identifiers (e.g. age and weight). However, to achieve this, the original data has to be modified. Such changes have to be minimized in order to facilitate future analysis.

In this project, we have implemented 3 distributed algorithms using Spark which aim to achieve k-anonymity while minimizing the changes on the data. Then, We have also evaluated their performance..

## Problem

In this project, we dealt with the dataset *data10k_6attr.csv* which consists of 10,000 data. Each data consists of 6 identifying attributes (age, height, weight, blood sugar level, number of children and weekly exercise hours) and a single sensitive attribute (risk of having disease). We anonymize the dataset so that for each combination of identifying attributes, there are at least k records with distinct sensitive attributes. The following figure shows an example for 4-anonymity.


After anonymization

| identifying attributes | | | | | | sensitive attribute |
|---|---|---|---|---|---|---|
| 19 | 146 | 56 | 3 | 2 | 1 | 3 |
| 28 | 163 | 67 | 8 | 5 | 13 | 3 |
| 33 | 176 | 78 | 6 | 4 | 6 | 1 |
| 32 | 130 | 68 | 8 | 4 | 2 | 3 |
| 82 | 137 | 46 | 21 | 4 | 11 | 4 |
| 28 | 146 | 62 | 16 | 0 | 11 | 5 |

| identifying attributes | | | | | | sensitive attribute |
|---|---|---|---|---|---|---|
| 30 | 150 | 62 | 4 | 4 | 7 | 3 |
| 30 | 150 | 62 | 4 | 4 | 7 | 3 |
| 30 | 150 | 62 | 4 | 4 | 7 | 1 |
| 30 | 150 | 62 | 4 | 4 | 7 | 3 |
| 82 | 137 | 46 | 21 | 4 | 11 | 4 |
| 28 | 146 | 62 | 16 | 0 | 11 | 5 |

The performance of k-anonymization is evaluated by calculating the data changes using the following function. The smaller changes an algorithm can achieve, the better performance it has.

$$changes = \sum_{i=1}^{n} |x_i' - x_i|$$

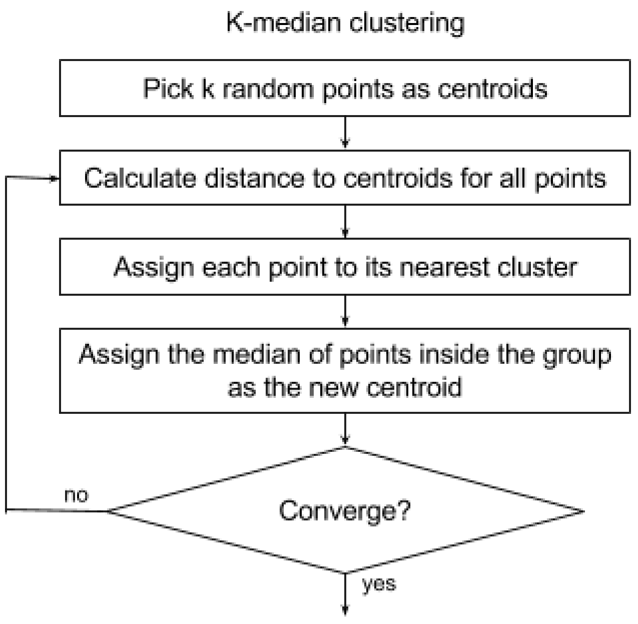
where n is the total number of data, $x_i$ and $x_i^{'}$ is the data before and after anonymization respectively.

## Algorithms

We have selected 3 clustering algorithms, which are K-medians clustering, Bisecting k-medians clustering and Density-based spatial clustering of applications with noise (DBSCAN), to anonymize the dataset.

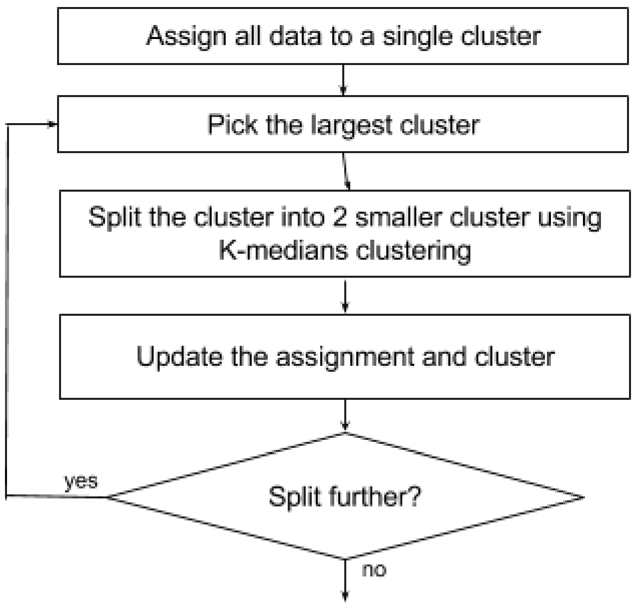
## K-medians clustering

K-medians clustering is a classic and popular clustering algorithm which takes a parameter k as the number of cluster it is going to construct. Then it creates k clusters and assigns each data point to its nearest cluster. The following figure shows the high level steps of the algorithm.


K-median clustering
Pick k random points as centroids
Calculate distance to centroids for all points
Assign each point to its nearest cluster
Assign the median of points inside the group as the new centroid
Converge?
no
yes

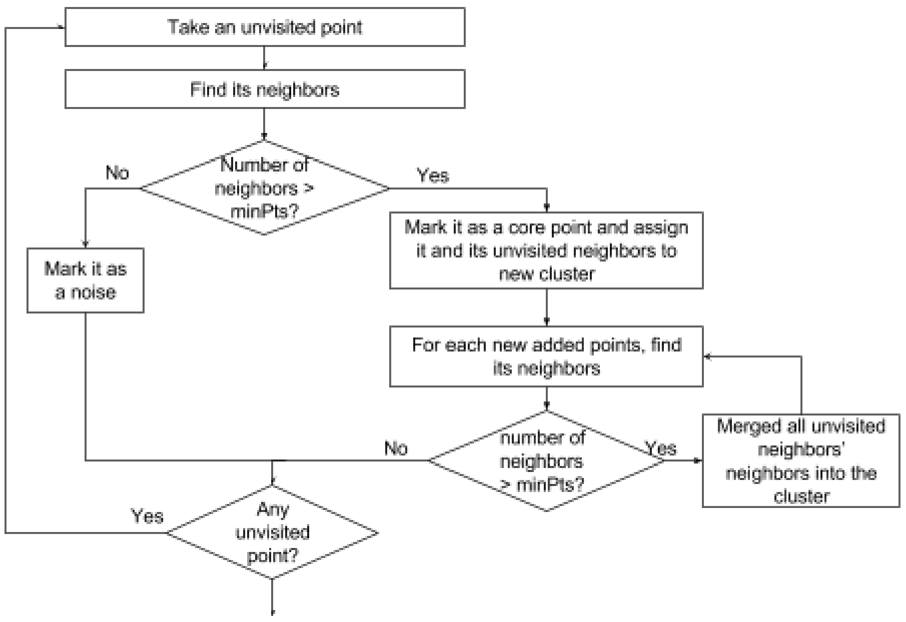
## Bisecting k-medians clustering

Bisecting k-medians clustering is a variant of K-medians clustering. Instead of configuring a particular number of cluster in advance, it keeps splitting the current clusters until no more split can be done. It consists of the following high level steps.


Assign all data to a single cluster
Pick the largest cluster
Split the cluster into 2 smaller cluster using K-medians clustering
Update the assignment and cluster
Split further?
yes
no

### Density-based spatial clustering of applications with noise (DBSCAN)

DBSCAN takes maximum radius ε (eps) and the minimal points required in the region of ε minPts. It forms clusters according to the density of given ε and minPts. The following figure shows the high level steps.


Take an unvisited point
Find its neighbors
Number of neighbors > minPts?
No
Yes
Mark it as a noise
Mark it as a core point and assign it and its unvisited neighbors to new cluster
For each new added points, find its neighbors
number of neighbors ≥ minPts?
No
Yes
Merged all unvisited neighbors' neighbors into the cluster
Yes
Any unvisited point?

## Implementation

As the original idea of the three algorithm is to assign data to the nearest cluster to reduce costs, it does not guarantee that each cluster has at least k members. To address the problem, we have made some modifications to the algorithms in order to achieve k-anonymity.

### K-medians clustering

To ensure k-anonymity, the algorithm ensures each cluster has at least k members after assigning all points to their nearest cluster. If a cluster has less than k members, it takes all the assigned members and asks for more in the next iteration. Clusters having more than k members must release the excess ones for others in the next iteration. After all clusters got at least k members, the algorithm assigns the remaining points to their nearest cluster.

To discover a minimum changes dataset, the algorithm tries different number of clusters instead of a particular number. It takes $\frac{n}{k}, \frac{n}{k} - 1, \frac{n}{k} - 2, ...$ clusters in order because more clusters has higher potential to achieve lower changes.

### Bisecting K-medians clustering

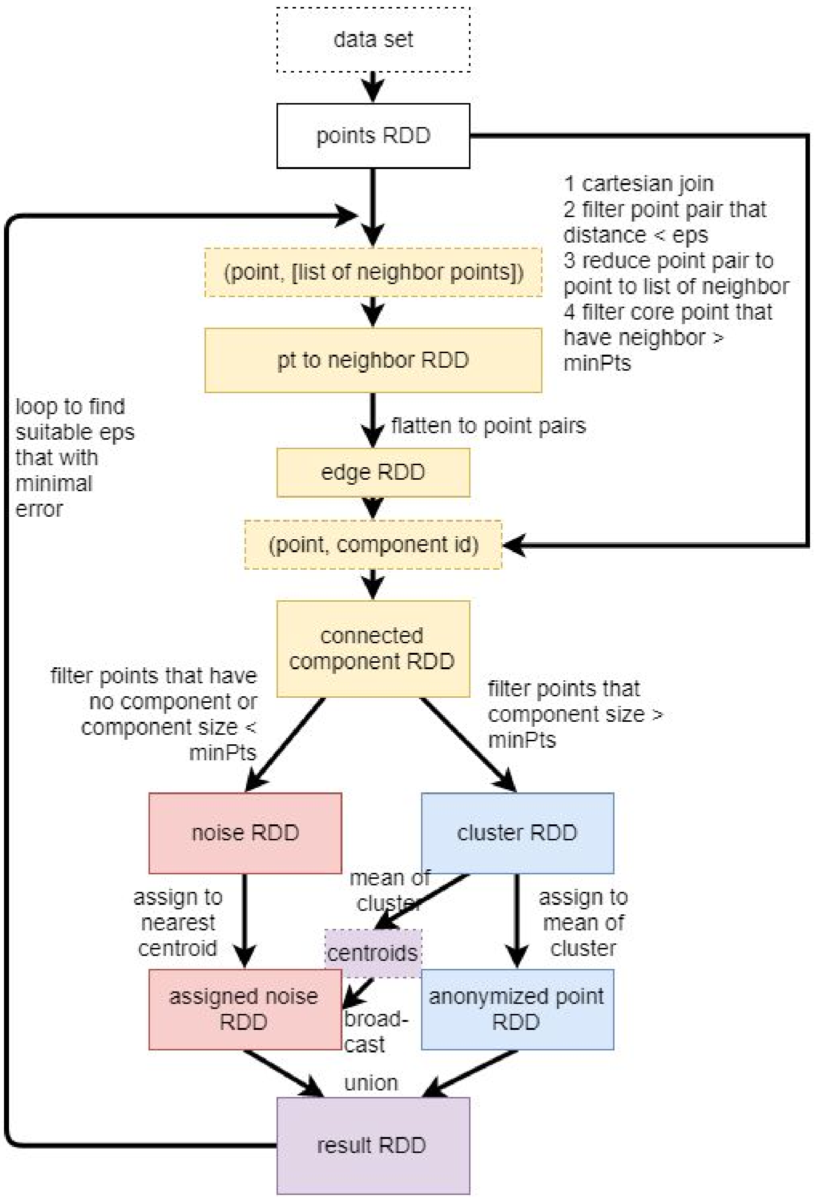
To ensure k anonymity, the algorithm first decides whether split a cluster or not by checking if the cluster contains at least 2k distinct members. If it decided to split, it applies the above modified K-median to split the cluster into two to ensure each of the splitted clusters also contains at least k members.

## DBSCAN

As original DBSCAN does not process noises (outliers that are not reachable by any core points within certain distance), we assigned noises to their nearest cluster to achieve k-anonymity.

Also, choosing a good value of ε is very important as the resulting cluster and the total error is very sensitive to ε it. Unfortunately, it is hard to find the best value for ε because it depends on the real density of the data and the target k. Indeed, unlike the number of cluster for K-medians clustering, the value of ε and the step size are not limited to an integer. To address the issue, we set the range and step size manually after trial and errors. finally, we chose the minimal step size to be 1.

The following diagram shows the flow of our modified version of DBSCAN using Spark.



data set
points RDD
1 cartesian join
2 filter point pair that distance < eps
3 reduce point pair to point to list of neighbor
4 filter core point that have neighbor > minPts
(point, [list of neighbor points])
pt to neighbor RDD
loop to find suitable eps that with minimal error
flatten to point pairs
edge RDD
(point, component id)
connected component RDD
filter points that have no component or component size < minPts
filter points that component size > minPts
noise RDD
cluster RDD
mean of cluster
assign to nearest centroid
centroids
assign to mean of cluster
assigned noise RDD
broad-cast
anonymized point RDD
union
result RDD

## Results

We implemented the three modified algorithms using Spark and deployed them in a cluster using Amazon Web Services. The cluster consists of 10 m4.xlarge machines which include one master node and nine slave nodes.

As the final output of K-medians clustering and Bisecting k-medians clustering depend on the initial random choice of the centroids, they are not deterministic. Thus having them to exhaustive search for a minimized output is time consuming. Instead of waiting them to finish, we executed the two algorithms for multiple times with different number of cluster to find a compromised output.

Best results we obtained for each algorithm:

| **Algorithm** | **Anonymized output** | **Minimum changes** | **Number of clusters** | **Running time** |
|---|---|---|---|---|
| K-medians clustering | *k_median.csv* | 121922.0 | 998 | 3 hours (terminated) |
| Bisecting k-medians clustering | *bis_k_median.csv* | 258974.0 | 78 | 3 hours (terminated) |
| DBSCAN | *dbscan.csv* | 271536.1 | 50 | 35 mins |

From the above table, we can see that DBSCAN outperforms with the other two algorithms in terms of running time. As K-medians clustering and Bisecting k-medians clustering will run much longer than 3 hours if we did not terminate it, DBSCAN is indeed faster than five times of K-medians and Bisecting k-medians. However, K-medians clustering achieved the minimum changes among three algorithms. The minimum changes made by K-medians clustering was half of the other two.

## Conclusion

In this project, we implemented algorithms to change the original data directly which may not be practical in real life. To apply these algorithms in real world problem, many alternative modification can be taken to preserve more information such as grouping a range of age as [XX-YY] is more informative than directly replacing all data with a single value. Although additional works have to be done for real applications, this project demonstrated the possibility and performance of data anonymization using clustering algorithm in cloud.