DIFFERENTIALLY PRIVATE FRAMEWORK FOR CROWDSOURCED ROAD MONITORING APPLICATIONS

By

SWARNAVO NATH

Class Roll No: 001910503047

Examination Roll No: MCA226046

Registration No: 149907 of 2019-2020

Master of Computer Application(MCA)
Computer Science and Engineering Department

Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

This is to certify that the project/thesis entitled "DIFFERENTIALLY PRIVATE FRAMEWORK FOR CROWDSOURCED ROAD MONITORING APPLICATIONS" has been satisfactorily completed under my guidance and supervision by **SWARNAVO NATH** (University Registration No.:**149907 of 2019-2020**, Examination Roll No.: **MCA226046**, Class Roll No.: **001910503047**). I hereby recommend that the project be accepted in partial fulfilment of the requirement for the Degree of Master of Computer Application, Department of Computer Science and Engineering in Faculty of Engineering and Technology, Jadavpur University, Kolkata for the academic year 2021-2022.

Dr. Chandreyee Chowdhury (Supervisor)
Associate Professor
Department of Computer Science and Engineering, Jadavpur University,
Kolkata-700032

Countersigned

Prof. Anupam Sinha
Head, Department of Computer Science and Engineering,Jadavpur University, Kolkata-700032.

Prof. Chandan Mazumdar
Dean, Faculty of Engineering and Technology,Jadavpur University, Kolkata-700032.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

This is to certify that the project/thesis entitled "DIFFERENTIALLY PRIVATE FRAMEWORK FOR CROWDSOURCED ROAD MONITORING APPLICATIONS" is a bona fide record of work carried out by SWARNAVO NATH in partial fulfilment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University during the period of January 2022 to June 2022. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

_____

Signature of Examiner
Date:

_____

Signature of Supervisor
Date:

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this project/thesis entitled "DIFFERENTIALLY PRIVATE FRAMEWORK FOR CROWDSOURCED ROAD MONITORING APPLICATIONS" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Application. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: SWARNAVO NATH
University Registration No.: 149907 of 2019-20
Examination Roll No. : MCA226046

Project/Thesis Title: DIFFERENTIALLY PRIVATE FRAMEWORK FOR CROWDSOURCED ROAD MONITORING APPLICATIONS

_____

Signature
Date:

ACKNOWLEDGEMENT

I express my deep sense of gratitude to my respected and learned guide, Dr. CHANDREYEE CHOWDHURY for her valuable help and guidance. I am grateful to her for the encouragement she has given me in completing the project.

SWARNAVO NATH
Roll No.: 001910503047
Examination Roll No.: MCA226046
University Registration No.: 149907 of 2019-2020

Table of Contents

Chapter 1: Introduction

1.1. What is Crowdsourcing?

The term 'crowdsourcing' first appeared in a **Wired article by Jeff Howe and Mark Robinson** in 2006. [1] The word highlighted a new way of connecting with people willing to work collaboratively on a project.

Crowdsourcing is the process of collecting services, ideas or content through the contributions of a large group of people. Usually, the "crowd" in crowdsourcing is a third-party unrelated to the business seeking results. One may crowdsource insights from customers and online communities, rather than employees or shareholders. [2]

John Howe published a definition for the term 'crowdsourcing' in a blog post: [3]

"Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers."

According to Daren C. Brabham, crowdsourcing can be **defined as an "online, distributed problem-solving and production model."** [4] [5]

Kristen L. Guth and Brabham found that **the performance of ideas offered in crowdsourcing platforms are affected not only by their quality, but also by the communication among users about the ideas, and presentation in the platform itself**. [6]

Having studied more than 40 definitions of crowdsourcing in scientific and popular literature, Enrique Estellés-Arolas and Fernando González Ladrón-de-Guevara, researchers at the Technical University of Valencia, defined crowdsourcing as: [7]

"Crowdsourcing can either take an explicit or an implicit route. Explicit crowdsourcing lets users work together to evaluate, share, and build different specific tasks, while implicit crowdsourcing means that users solve a problem as a side effect of something else they are doing. With explicit crowdsourcing, users can evaluate particular items like books or webpages, or share by posting products or items. Users can also build artifacts by providing information and editing other people's work. Implicit crowdsourcing can take two forms: standalone and piggyback. Standalone allows people to solve problems as a side effect of the task they are doing, whereas piggyback takes users' information from a third-party website to gather information."

1.2. Types of Crowdsourcing

Crowdsourcing can be either of explicit or implicit type: [1]

In case of **Explicit crowdsourcing**, users work together to evaluate, share, and build different specific tasks. They can evaluate particular items like books or webpages, or share by posting products or items. Users can also build artifacts by providing information and editing other people's work.

Implicit crowdsourcing can be of two types: standalone and piggyback. Standalone allows people to solve problems as a side effect of the task they are actually doing, whereas piggyback takes users' information from a third-party website to gather information.[8] This is also known as data donation.

Daren C. Brabham puts forth **a problem-based typology of crowdsourcing approaches** in his 2013 book, 'Crowdsourcing': [9]

a) Knowledge discovery and management is used for information management problems where an organization mobilizes a crowd to find and assemble information. It is ideal for creating collective resources.

b) Distributed human intelligence tasking (HIT) is used for information management problems where an organization has a set of information in hand and mobilizes a crowd to process or analyse the information. It is ideal for processing large data sets that computers cannot easily do. Amazon Mechanical Turk uses this approach.

c) Broadcast search is used for ideation problems where an organization mobilizes a crowd to come up with a solution to a problem that has an objective, provable right answer. It is ideal for scientific problem-solving.

d) Peer-vetted creative production is used for ideation problems, where an organization mobilizes a crowd to come up with a solution to a problem which has an answer that is subjective or dependent on public support. It is ideal for design, aesthetic, or policy problems.

Crowdsourcing based on How Various Applications Function:[10]

Different types of Crowdsourcing can also be categorized by analysing how various crowdsourced applications function. Using this approach, **Jeff Howe further classified crowdsourcing in the following four categories: [10]**

a) **Crowd wisdom** – This technique uses the "collective intelligence" of people within or outside an organization to solve complex problems. 'Innocentive' is one such example

b) **Crowd creation** – It leverages the ability and insights of a crowd of people to create new products. I love Quirky's co-creation community is one such example.

c) **Crowd voting** –The community votes for their favourite idea or product. Threadless is one such example given by Howe himself.

d) **Crowd funding** – There is a proliferation of different types of crowdfunding platforms in the market (rewards-based such as Kickstarter and equity-based such as CrowdCube) and serving different purposes.

1.3. Examples of few popular Crowdsourced Applications

1.3.1. Wikipedia

This is the go to website for all of us whenever we need information about any topic. It is a free online encyclopedia written and maintained by a community of volunteers through open collaboration and a wiki-based editing system. While Wikipedia is capable of providing information about almost everything in this world, all the information that Wikipedia contains is actually collected via crowdsourcing. Steven Pruitt is one such Wikipedia contributor who has made highest number of Wikipedia edits at over 4.7 million.

1.3.2. Amazon Mechanical turk(MTurk)

Amazon Mechanical Turk (MTurk) is a crowdsourcing website for businesses (known as Requesters) to hire remotely located "crowdworkers" to perform discrete on-demand tasks that computers are currently unable to do. It is operated under Amazon Web Services, and is owned by Amazon. Employers post jobs known as Human Intelligence Tasks (HITs), such as identifying specific content in an image or video, writing product descriptions, or answering questions, among others.

1.3.3. Stack Overflow

Stack Overflow is a question and answer website for professional and enthusiast programmers. It features questions and answers on a wide range of topics in computer programming. While one may find solutions to the programming problems that s/he is facing on this site, the solution is actually provided by other Stack Overflow users. This is a classic and popular example of crowdsourcing.

1.3.4. Quora

Quora is another very popular crowdsourced website where the questions posted by one user are answered by other Quora users.

1.4. Advantages and Disadvantages of Crowdsourcing

1.4.1. Advantages

The advantages of crowdsourcing include cost savings, speed, and the ability to work with people who have skills that an in-house team may not have. If a task typically takes one employee a week to perform, a business can cut the turnaround time to a matter of hours by breaking the job up into many smaller parts and giving those segments to a crowd of workers.

Many types of jobs can be crowdsourced, including website creation and transcription. Companies that want to design new products often turn to the crowd for opinions. Rather than rely on small focus groups, companies can reach millions of consumers through social media, ensuring that the business obtains opinions from a variety of cultural and socioeconomic backgrounds. Oftentimes, consumer-oriented companies also benefit from getting a better gauge of their audience and creating more engagement or loyalty.

1.4.2. Disadvantages

However, this 'Crowdsourcing' concept is not something which does not have any shortcomings.

1.4.2.1. Lack of user interest

The success of any crowdsourced idea/task depends on the successful participation of people who actually contribute towards solving the task in hand. However, if there is no interest or benefit provided to these people, as a natural effect there will be less participation. So, businesses and organizations should first understand the user sentiment and accordingly decide upon providing various social and financial (incentives) benefits to the contributors in order to encourage them to participate in the crowdsourcing process.

1.4.2.2. Reliability of the information provided by contributors

Even after mobilizing sufficient number of contributors by providing them the desired benefits, there is always uncertainty about the correctness and reliability of the content that the contributors are providing. Result of any crowdsourced work can be skewed easily based on the crowed being sourced. So, identifying the correct crowd is also a major challenge for crowdsourcing.

1.4.2.3. Privacy of contributors

Even if we can mobilize the right people by giving them the desired benefits for their contribution, the next major concern is the privacy of these contributors. Depending upon the nature of a crowdsourced application, contributors might have to provide a lot of sensitive information about themselves. This information may include their personal, professional life, their opinion on various social/economical/political topics, their likes and dislikes on various matters. All of this information can be further used to personally identify, track and understand the contributors which possess a severe privacy risk in any crowdsourced applications.

This leads to the motivation of our work, i.e. how to protect the anonymity of contributors in crowdsourced applications which is discussed in section 1.5.

1.5. Motivation

Crowdsourcing can be proved to be very effective in a Road monitoring application. Only knowing the shortest route to reach the destination is not enough while travelling. There are several other common important factors that people usually look for while deciding on which route to travel before they actually make the journey. For example,

i) A route where the **Road Condition** is not so good is less likely to be taken by an ambulance or a pregnant woman, even if it's the shortest route to reach to destination.

ii) During monsoon, people usually won't prefer to travel in a route which has **Water Logging** issue.

iii) A route which is usually very busy and has high **Traffic Congestion** will not be preferred by the commuters who are in hurry.

iv) A route which is **mostly deserted is less likely to be taken by a woman** at night even if it is the shortest one to reach her destination.

v) When a long distance has to be covered and the driver is not sure about the fuel consumption that it will take, **a route which has very less number of gas stations in between source and destination** is less likely to be taken.

Considering all these factors, we have built this '**Crowdsourced Road Monitoring Application**' where a user can review the routes that s/he has travelled based on these parameters. This route information provided by one user can eventually help other users to decide which route to choose based on their needs and preferences.

However, as discussed in section 1.4.2.3, the major challenge after building one such crowdsourced road monitoring application is to protect the privacy of contributors reviewing the routes. While the nature of this application demands the review list of a user to be made public, under no circumstances we can allow an adversary to guess the whereabouts (such as residence, workplace) of our contributors by seeing the routes that s/he has reviewed.

This privacy challenge motivates us to further implement **Differential Privacy** in our Crowdsourced Road Monitoring application in order to protect anonymity of our contributors. In section 5.2.4 and 5.2.5, we have further discussed in detail about when and how to introduce noise in our dataset in order to make it differentially private.

1.6. Contribution

A Crowdsourced Road Monitoring Application has been developed first in order to collect data from users. Then Differential Privacy has been implemented on the dataset that are collected using our Crowdsourced Road Monitoring Application.

1.7. Organization of this report

1.7.1. Related Work

This section refers to some of the similar types of works on implementing Differential Privacy in Crowdsourced applications. It will help us understand the uniqueness of the work that we have done in this project.

1.7.2. Differential Privacy

This section explains the concept of Differential privacy, mathematical equations and the scenarios in which it can be implemented. The challenges and limitations of Differential privacy are also discussed.

1.7.3. Proposed Methodology

It presents a brief overview of the main idea behind this project, what we are planning to achieve and the proposed methodology in order to achieve it.

1.7.4. Implementation Details

In this section we have illustrated how the idea has actually been implemented. The technology stack which has been used for implementation of our idea is mentioned here. The algorithms that we have developed to achieve our desired objective are explained in great detail with the help of flowcharts and code snippets.

1.7.5. Experimental Results

This section helps us understand how much successful we have been in order to implement our idea. With the help of different types of graphs made from the dataset of different users, we have analyzed how effective the implementation of differential privacy has proven to be. We have also talked about the data loss and corresponding loss in accuracy of information as an adverse effect of implementing differential privacy.

Chapter 2: Related Work

2.1. Deep Learning with Differential Privacy

Neural networks based Machine learning techniques are achieving remarkable results in a wide range of domains.

In the work [11], Abadi et al. developed new algorithmic techniques for learning and a refined analysis of privacy costs within the framework of differential privacy in crowdsourced large datasets containing sensitive user information, which are often required for training of various machine learning models. In this paper, the authors combine state-of-the-art machine learning methods with advanced privacy-preserving mechanisms, training neural networks within a modest ("single-digit") privacy budget. Models are treated with non-convex objectives, several layers, and tens of thousands to millions of parameters. The author demonstrates that much tighter estimates on the overall privacy loss, both asymptotically and empirically can be obtained by tracking detailed information of the privacy loss. The computational efficiency of differentially private training can be improved by introducing new techniques such as efficient algorithms for computing gradients for individual training examples, sub dividing tasks into smaller batches to reduce memory footprint, and applying differentially private principal projection at the input layer. The author evaluates his approach on two standard image classification tasks, MNIST and CIFAR-10 and claims to achieve 97% training accuracy for MNIST and 73% training accuracy for CIFAR10, both with differential privacy.

2.2. Privacy-Preserving Traffic Volume Estimation by Leveraging Local Differential Privacy

In the work [12], Oh et al. present a method for effectively predicting traffic volume based on vehicle location data that are collected by using LDP (Local Differential Privacy). The proposed solution in this paper consists of two phases: the process of collecting vehicle location data in a privacy-preserving manner and the process of predicting traffic volume using the collected location data. In the first phase, the vehicle's location data is collected by using LDP to prevent privacy issues that may arise during the data collection process. LDP adds random noise to the original data when collecting data to prevent the data owner's sensitive information from being exposed to the outside. This allows the collection of vehicle location data, while preserving the driver's privacy. In the second phase, the traffic volume is predicted by applying deep learning techniques to the data collected in the first stage.

2.3. Defeating Traffic Analysis via Differential Privacy: A Case Study on Streaming Traffic

The authors Zhang et al. in their work [13] explore the adaption of techniques previously used in the domains of adversarial machine learning and differential privacy to mitigate the ML-powered analysis of streaming traffic. The findings are twofold. First, constructing adversarial samples effectively confounds an adversary with a predetermined classifier but is less effective when the adversary can adapt to the defence by using alternative classifiers or training the classifier with adversarial samples. Second, differential privacy guarantees are very effective against such statistical-inference-based traffic analysis, while remaining agnostic to the machine learning classifiers used by the adversary. For enforcing differential privacy for encrypted streaming traffic and evaluating their security and utility, the authors propose three mechanisms:

(i) Fourier Perturbation Algorithm (FPAk) [14], a differentially private mechanism that answers long query sequences over correlated time series data in a differentially private manner based on the Discrete Fourier Transform (DFT)

(ii) d$*$-private mechanism (d$*$), which extends the d-privacy mechanism from Chan et al. [15] and applies Laplacian noise on time series data

(iii) dL1-private mechanism (dL1), which achieves d-privacy with regard to L1 distance.

2.4. Privacy-Preserving Data Collection Scheme on Smartwatch Platform

In recent years, smartwatches, which are the most representative wearable device exploiting Internet of things technologies, have been developed and used. With growing use of smartwatches in the healthcare field, there have been considerable efforts to use diverse smartwatch sensor data such as heart rates and body temperatures with the aim of improving healthcare services. However, because the data collected from smartwatches usually contain sensitive user information, individual users are reluctant to provide their data to healthcare service providers because of privacy concerns.

In the work [16] Kim et al. propose a method capable of collecting individuals' sensitive data from smartwatches, while preserving privacy, in order to meet the needs of healthcare service providers. The proposed method in this study uses LDP to collect sensitive personal data from smartwatches in a privacy-preserving manner. LDP ensures that the data contributor's original data are not exposed to the outside, because the data contributor first perturbs the original data by adding carefully designed noises and then sends the noisy data to a data collector. Then, the data collector can compute population statistics based on the large volume of the noisy dataset.

2.5. Evaluating Machine Learning Models for Handwriting Recognition-based Systems under Local Differential Privacy

Handwriting recognition, a pervasive assistive technology for a variety of consumer electronics and Internet of Things (IoT) systems does not provide meaningful privacy guarantees. Hacking of a device, malware, and compromise of handwriting recognition-enabled application's cloud service can lead to leakage of sensitive personal information with serious consequences.

In [17], Shahid et al. propose an algorithm to deform a user's original handwriting by adding adjustable statistical noise before sharing it with a specific application to achieve local differential privacy (LDP). To assess and quantify the algorithm's impact on recognition, the authors generate a shape-deformed test set by applying the algorithm to the MNIST dataset and implements a visualization tool to demonstrate the impact of the proposed algorithm. Then, the authors train a convolutional neural network, Naive Bayes, random forest, support vector machine, and decision tree classifiers on the MNIST dataset and evaluate them with the privatized test set and claims to achieve around 80% accuracy in local differential privacy settings.

Chapter 3: Differential Privacy

3.1. Understanding Differential Privacy [18]

Differential Privacy is a privacy preserving technique to share information publicly about a dataset (by identifying the patterns of groups within a dataset) while ensuring that information about any individuals in that dataset is withheld. The idea is to learn nothing about an individual while learning useful information about a population.

Differential privacy ensures that same conclusion will be reached always whether or not any individual opts into the dataset. If we perform any query on a differentially private dataset, we should not be able to infer anything about any individual in the dataset by seeing the query result. The idea is that it should not make any impact whether or not an individual is present in the dataset.

While publishing any aggregate information about a statistical database, a differentially private algorithm limits the extent to which (private) information from the database can be disclosed.

An algorithm is said to be differentially private if an observer seeing the output can never infer whether or not an individual's information is present in the input dataset which was used for computation. However, the Fundamental Law of Information Recovery states that overly accurate answers to too many questions will destroy privacy completely. The goal of algorithmic research on differential privacy is to delay this inevitability as long as possible.

So, differential privacy becomes essential while identifying individuals whose information may be present in a dataset. Differentially private algorithms are used to resist identification and re-identification attacks.

Scenarios where Differential Privacy can be implemented:

i) Government agencies that publish demographic information. Such agencies must always ensure confidentiality of the users who participated in the survey

ii) Companies that collect user information to understand the user behaviour must put a limit to what is visible even to the analysts

iii) Any crowdsourced applications collecting sensitive user data while ensuring that no personally identifiable information about any user should be visible to other users

3.2. Equation [19]

The mathematical definition of differential privacy [21] is stated as - a randomized algorithm L gives ε-differential privacy if for all data sets $| X - Y | \leq 1$ and any $S \subset$ Range(L), where S: All potential output of L that could be predicted.

$$Pr[L(X) \in S] \leq e^{\varepsilon} Pr[L(Y) \in S]$$

Here, the two factors that are used to measure differential privacy are as follows:

Accuracy is the closeness of the output of differential privacy algorithms to the original output.

Epsilon(ε) in differential privacy algorithms is an assessment of privacy loss at a differential changed data.

3.3. Challenges and Limitations of Differential Privacy

1. The idea of Differential Privacy can't be applied to every problem, such as,

a) Individual level analysis: Such analysis is not possible in a dataset which is differentially private. Since, differential privacy relies on the idea of learning nothing about an individual while learning useful information about a population irrespective of whether or not the individual is present in that population, it prevents analysts from learning specific or private information about any specific individual.

b) Small data: The noise or inaccuracy in a dataset introduced by applying differential privacy will be unnoticeable for large datasets, while it will have a big negative impact on the accuracy of the output and any analysis performed on the output dataset, if we tend to do the same for smaller datasets.

2. **Correct level of Ɛ is not clear**:

There is no standard way of calculating how much noise we should introduce in a dataset in order to make it differentially private while also maintaining the accuracy of the output. Higher the value of ε, it means the data is more accurate but less private. Therefore, ε = 0 should be the perfect privacy case ideally but it completely distorts the original data and makes it useless. So, deciding on the value of ε, which controls the trade-off between privacy and data utility is a major challenge while designing any differentially private algorithm.

Chapter 4: Proposed Methodology

4.1. Task Definition

The objective of this work is to preserve the privacy of the users in any crowdsourced applications. Any personally identifiable information of the contributors (in any crowdsourced application) should not be exposed to the outside world. So, our task here is to maintain the privacy of the users with minimal complexity and data loss so that the accuracy of user information is not compromised significantly.

For the demonstration, we have built a Differentially Private Crowdsourced Road Monitoring application where a user can review routes which he/she has travelled based on various parameters.

Input:

1. n number of routes reviewed by a particular user
2. m(m<=n) number of clusters each containing a number of routes. Either source or destination of all the routes belonging to a particular cluster is either same or they are closed to each other (i.e. within a certain distance)

Output:

1. Identifying the clusters which has significantly more number of route reviews than other clusters. This clear peak in the number of reviews for a particular cluster indicates that the user travels frequently in those areas which belong to that cluster. This information can eventually lead to compromise of user's location privacy.
2. Remove a few routes from the cluster which has significantly greater number of route reviews so that it doesn't have any clear peak after removing. In this way, if there is no clear peak in the number of reviews in any cluster, it will be less likely to accurately guess the areas where a particular user frequently travels in.

Objective:

Prevent any adversary to infer anything about the whereabouts of a user from the routes that s/he has reviewed.

4.2. Approach

4.2.1. Formation of Cluster

Firstly, all the routes rated by a user are decomposed into two parts based on its source and destination. Clusters are formed based on either source or destination. A cluster contains all those routes which has either same source or same destination or those routes for which the sources (or, destinations) are within a certain distance. An individual cluster contains the following information:

i) Basis (either source or destination) on which the cluster has been formed

ii) total number of routes which belong to that cluster

iii) all the routes' information which belong to the same cluster

For an individual route, if the clusters exist for both its source and destination, then between those two, it will be part of that cluster which has maximum number of routes. Thus, it is ensured that no single route is there in two different clusters, once based on its source and then based on its destination. So, there are no duplicate route reviews in any clusters.

Later in section 5.2.4, we have discussed in detail how the clusters are being formed.

4.2.2. Privacy Metric

Privacy metric is used to identify the clusters which has significantly more number of reviews than others and after selecting the cluster, it also determines which are the routes that should be removed so that the accuracy of user information is not compromised to a significant extent.

Step 1: Identifying the Clusters which are significantly bigger in size:

Calculate mean of total number of route reviews present in all the clusters.

Select those clusters for which the number of routes are greater than or equal to the value of outlier. Where, outlier = **(calculated mean + 25 % of calculated mean)**

Step 2: Routes which should be removed inside a cluster:

For a route, calculate its distance with all other routes belonging to the same cluster and then find the Standard Deviation of all the distances calculated. The route which has lowest standard deviation value indicates that all other routes are within close proximity of this particular route. Thereby, if this particular route is removed, the overall accuracy of the information is not drastically hampered. Therefore, the route with lowest standard deviation value inside a cluster should be removed first. And this removal process continues until the number of reviews inside that cluster is less than our value of outlier which has been calculated in step 1.

Implementation of this privacy metric is further discussed in detail in section 5.2.5.

4.2.3. Workflow Diagram

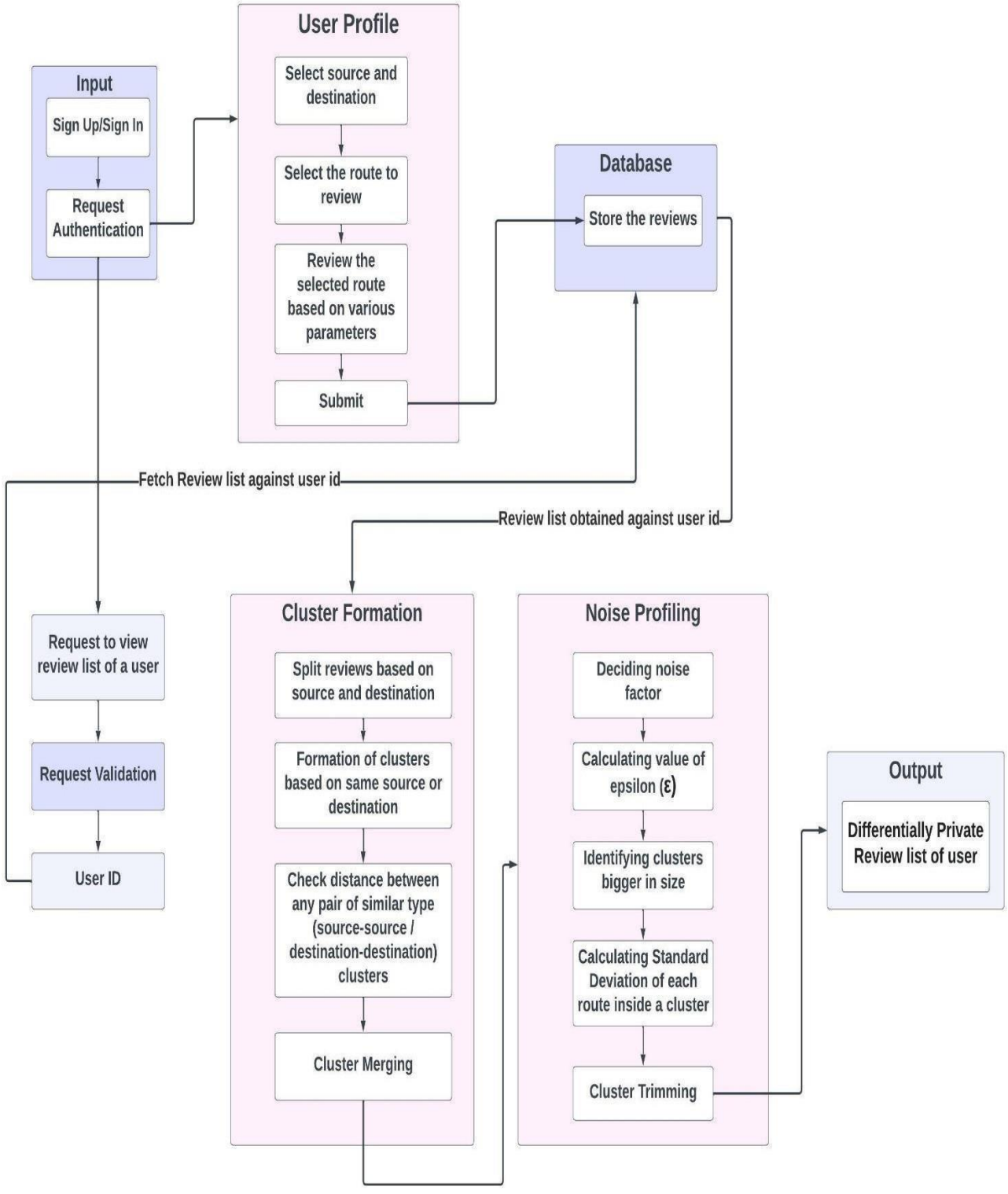


Fig 1: Workflow of the proposed Differentially Private Crowdsourced Road Monitoring Application framework

4.2.4. Parameters for Differential Privacy

Differential Privacy tackles the challenge when an adversary attempts to track one individual user and figure out his/her corresponding data. Having some prior knowledge about the user, the probability of finding the information is P(in) / P(out) of the user to be in the database than isn't and the measure of that is ε in differential privacy. So, w.r.t our problem, the data collected from our Crowdsourced Road Monitoring Application is taken as input and a privacy metric is applied on this dataset in order to make it differentially private. Thus, formulating from the basic equation of differential privacy mentioned in section 3.2, the differential privacy definition, $e^{\varepsilon} >= P(in) / P(out)$, where P(in) in our case is $P(\text{Z-Score}_{\text{real data}})$ which is calculated from the Z-table and P(out) is $P(\text{Z-Score}_{\text{perturbed data}})$ calculated from Z-table.

Therefore, ε is calculated as follows:

$$\varepsilon = \ln (P(\text{Z-Score}_{\text{real data}}) / P(\text{Z-Score}_{\text{perturbed data}}))$$

Z Score is calculated as follows:

Z-Score = $(X - \mu) / \sigma$

Where, X = Highest value (element) present in cluster distribution of our dataset

μ = Mean value of cluster distribution of our dataset

σ = Standard Deviation of cluster distribution of our dataset

P(Z-Score) is calculated from the Z-Table.

4.2.5. Algorithm for Clustering

1. Fetch Review List of user from database.
2. Iterate through the review list and create two map objects:
   Source as key with array of reviews having same source against it and destination as key with array of reviews having same destination against it.
3. Extract key sets of both the maps and store it into two separate arrays.
4. Sort the two arrays in the descending order of the length of review arrays in the maps
5. Iterate both the arrays containing the source and destination keys
6. Check if elements at current index in both the arrays has equal number of reviews
   - If yes, increment the index and check again
   - If no, go to step 7.
7. Check if size of review array is bigger against source:
   - If yes, Review list against the key in source map from index 0 up to current index is clubbed into single cluster
   - If no, Review list against the key in destination map from index 0 up to current index is clubbed into single cluster
8. The original clusters are modified
9. Check if further clustering is required:
   - If yes, go to step 2
   - If no, merge any two source/destination clusters which are within a certain distance

Chapter 5: Implementation Details

For the demonstration purpose, we have built a Crowdsourced road rating and monitoring application where the users can select a route and provide ratings as per their preference based on various parameters such as Road condition, women safety, traffic congestion, water logging and frequency of gas station.

5.1. Technologies Used

Front End: To build the user interface, we have used a very popularly used Javascript framework named React JS.

Back End: On the server side, another commonly used Javascript framework Node JS has been used.

Database: To store any data, we have used a No-Sql database named Mongo DB.

Deployment: The application has been deployed in a cloud application platform named Heroku.

5.2. Functionalities of the application

5.2.1. Register/Login Functionality

When a user visits our web application for the first time, s/he has to register first using his name, email id and password. If, the user is already existing, s/he can directly login to his/her account using appropriate email-id and password. It is mandatory for the users to create an account first. Without having one, no users can use this application.

5.2.2. Submitting reviews for any route

Once the user is logged into his/her account, s/he will be displayed the home page, where the user can select a pair of source and destination and click on the 'Get Route' button to get the route direction. These source and destination locations and the corresponding route information have been obtained using the MapQuest Developers' API. As per our implementation, for a particular <source-destination> pair, the number of routes which will be displayed has been limited to 2 routes.

Once the routes are displayed on the browser, the user can select a route (the selected route will appear as a blue line in the map and the alternate route will appear as a pink line. The user has to click on a route in order to select it) in order to review it based on various parameters. It will take the user to a different page where the user can provide his rating based on five different parameters which are:

a) Road Condition

b) Traffic Congestion

c) Women Safety

d) Water Logging

e) Frequency of Gas station

5.2.3. Viewing all the routes reviewed by a user

On the home page, if the user clicks on the 'My Reviews' option displayed at the top right corner of the screen, s/he will be redirected to a new page which will display a list of all the route reviews that have been submitted by him/her so far.

Here, each route review contains the following information:

a) From: From location indicates the source from where the user has started the journey.

b) To: To location indicates the destination of a journey

c) Path Latitudes: This contains the latitude points of locations between source and destination.

d) Path Longitudes: This contains the longitude points of the locations between the source and destination.

These latitude and longitude points are obtained from MapQuest API which has been called when the user clicks on the 'Get Route' option after entering his/her preferred source and destination locations and these latitude-longitude points have been further used in section 5.2.5 (Privacy module) to calculate the distance between any two routes.

e. Reviews: It contains an array of 5 integers where each integer value corresponds to the rating of a particular review parameter.

First integer value corresponds to the rating of Road Condition

Second value corresponds to the rating of Women Safety

Third value corresponds to the rating of Water Logging (Higher the rating, lesser the water logging)

Fourth value corresponds to the rating of Traffic Congestion (Higher the value, lesser the congestion)

Fifth value corresponds to the rating of Frequency of gas station between source and destination.

f. Created At : It indicates the timestamp when the review was submitted by the user.

5.2.4. Formation of Cluster (Identifying the routes which are close to each other)

Since, our main goal is to protect users' location privacy while a user is submitting reviews for the routes that s/he has travelled in, we have to identify the routes rated by a user which are nearby to one another. If a user has rated too many routes whose sources or destinations are either same or within a very close distance of one another, by seeing those reviews one may infer the whereabouts of the user. This can eventually lead to the potential loss of user privacy.

In order to identify such routes which are close to each other, we form the clusters here. A cluster is formed either based on source of a route or destination. If a cluster is formed on the basis of source, then all other routes inside that cluster will have either same source or the sources are within a certain distance. As we have the latitude-longitude points arrays for all sources and destinations, we calculate the distance using Haversine Distance formula between a pair of sources or destinations. In our case, the threshold value for this distance parameter is taken to be 3 kilometer. It means if two routes have either same sources or two different sources which are within 3 kilometer distance of one another, those two routes will belong to the same cluster. The logic is same while the clustering is being done on the basis of destination of a route.

5.2.4.1. Cluster formation based on same source/destination

Here, with the help of flowcharts we will discuss in detail how clusters are formed if the routes are having either same source or same destination, when the clustering should be based on the source of a route and when it should be on the basis of destination.

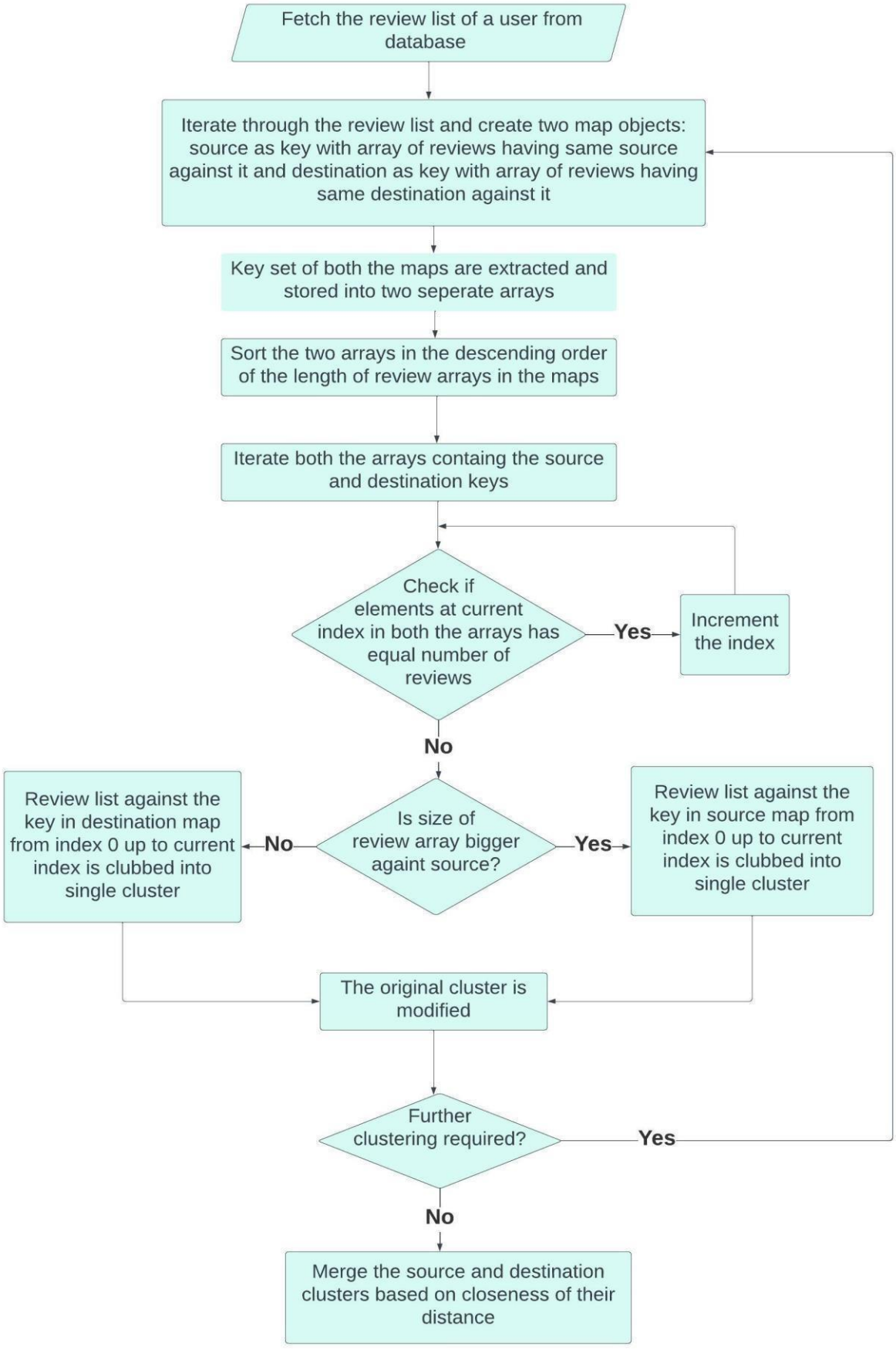


Fig 2: Formation of Clusters based on same source or destination

Steps (As shown in Fig 2):

i) A HTTP GET Request has been sent to the server to fetch all the reviews submitted by an individual user from our database.

Code Snippet:

```
const review=await axios.get(`http://localhost:5000/api/v1/reviews`
```

ii) We create two map objects, one for sources of all the routes, another for destinations. Now, we iterate through the review list obtained from the database one by one and decompose each route in the review list into two parts based on its source and destination. For a particular route, we check if its source and destination are already present as one of the keys in our maps.

If yes, that particular route is stored against its corresponding source key in the source map into an array and the same route is stored in the destination map against its corresponding destination key into an array.

If not, we first create a source key in the source map for that particular source and store the route against that source key and the same process is repeated for the destination map also.

Code snippet for the above logic is as follows:

```
const source={},dest={};
tempReviewList.map((item)=>{
  if(!source[item.from]){
     source[item.from]=[item]; }
  else{
     source[item.from].push(item); }
  if(!dest[item.to]){
     dest[item.to]=[item]; }
  else{
     dest[item.to].push(item); } })
```

iii) All the keys from the source map as well as destination maps are extracted and stored into two separate arrays. So, all the keys (sources) from the source map are stored in one array and all the keys (destinations) from the destination map are stored in another array.

Code snippet:

```
const sourceKey=Object.keys(source),destKey=Object.keys(dest);
```

iv) These two arrays are now sorted in the descending order of the number of routes that are there in the map for a particular source or destination. For a source S1 which is a key in the source map, if the source map contains highest number of routes for this key, then S1 will be the first element in the source array after sorting. This sorting logic is same for the other array also which stores all the destinations from the destination map.

Code Snippet:

```
sourceKey.sort((a,b)=>source[b].length-source[a].length);
destKey.sort((a,b)=>dest[b].length-dest[a].length);
```

v) Now, we iterate both the sorted source and destination arrays (indexing starts from 0) and check if the number of reviews for both the source and destination element at current index is equal or not.

If yes, we increment the index and perform the same checking for the elements at next indices until we find one such index where the number of reviews for both the corresponding source and destination present in that index is not same. If no such index is found, then the clustering will be based on source by default.

If not, we check whether the number of reviews is greater for source or destination at a particular index. If it is greater for source element, then from the initial index up to this particular index, the clustering will be done based on sources of the routes. Otherwise, it will be based upon destinations.

Code Snippet for the above logic is as follows:

```
function selectKeys(sourceKey,destKey,source,dest){
 for(let i=0;i<sourceKey.length && i<destKey.length;i++){
 if(source[sourceKey[i]].length>dest[destKey[i]].length){
     return {select:0,upto:i}; }
  else  if(source[sourceKey[i]].length<dest[destKey[i]].length){
    return {select:1,upto:i}; } }
  return {select:0,upto:sourceKey.length-1} }
```

vi) Now we decide whether further clustering is required or not. In order to that, all the routes which have been already included in any of the clusters formed are removed from the review list and we then check if our review list is empty or not.

If not, it means that these remaining routes in the review list yet don't belong to any of the clusters which have been formed so far. So, further clustering is required and we repeat the whole process from step i).

If yes, it means that all the routes which were there in the review list have already been included in any of the clusters formed. Therefore, there is no need for further clustering.

Code Snippet for the above logic is as follows:

```
for(let i=0;i<=selects.upto;i++){

clust.push({type:selects.select===0?"source":"destination",
```

```
            info:iterateMap[iterateKey[i]]});
iterateMap[iterateKey[i]].map((item)=>{
                removeList[item._id]=item;    })    }
tempReviewList=tempReviewList.filter((item)=>{
return !removeList[item._id] })
```

5.2.4.2. Merging the clusters based on closeness of distances between source or destination

In section 5.2.4.1, we have formed the clusters either based on same source or destination. All the routes inside a cluster have either same source or same destination.

Next, we check the distances between the sources in the source map and also, the distances between the destinations in the destination map using Haversine Distance formula.

Haversine's formula:

$a = \sin^2(\Delta latDifference/2) + \cos(lat1).\cos(lat2).\sin^2(\Delta lonDifference/2)$

$c = 2 * atan2(\sqrt{a}, \sqrt{1 - a})$

$d = R*c$

where,

$\Delta latDifference = lat1 - lat2$ (difference of latitude)

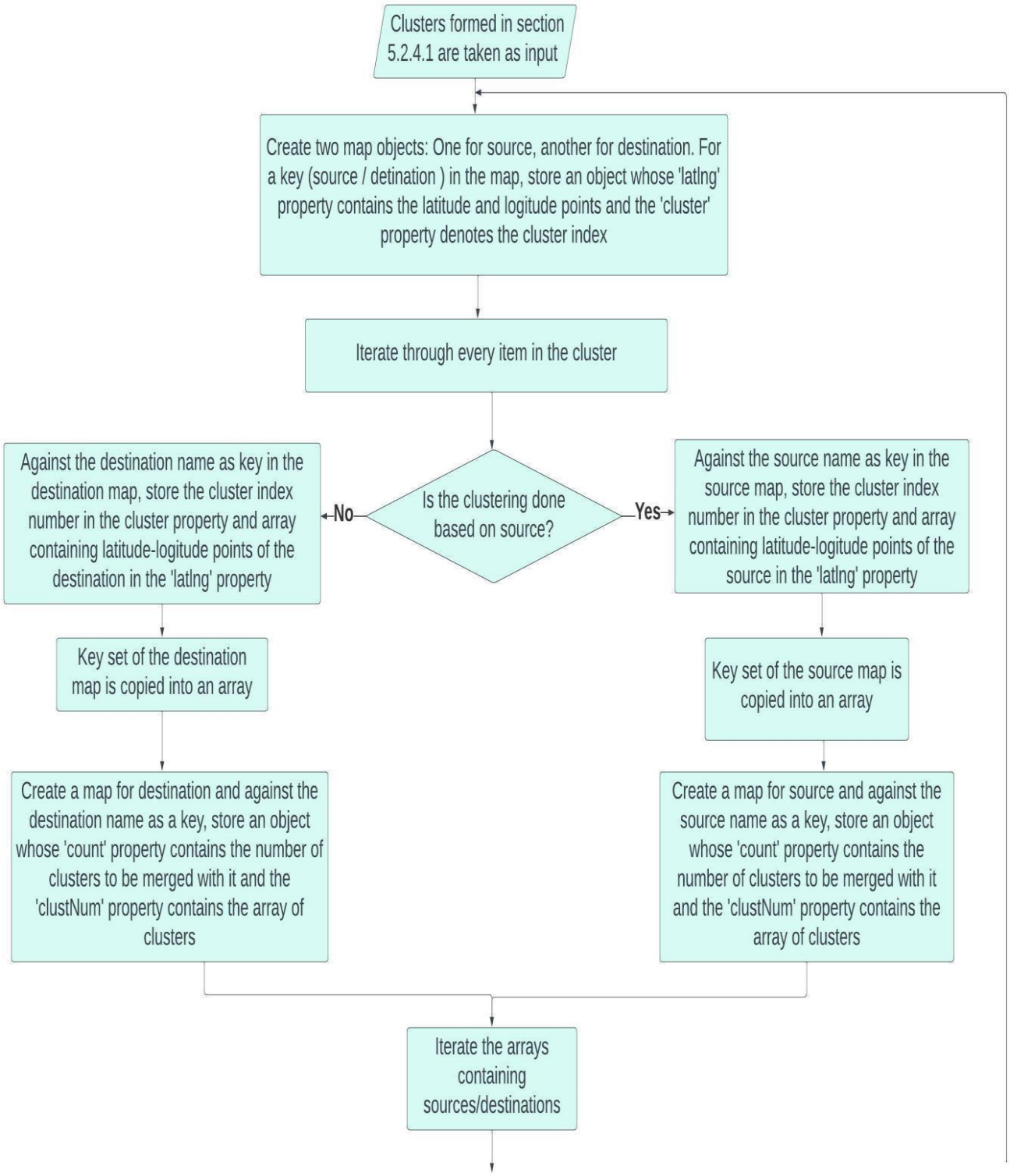
$\Delta lonDifference = lon1 - lon2$ (difference of longitude)

R is radius of earth i.e. 6371 KM or 3961 miles

and d is the distance computed between two points.

If any two sources or any two destinations are within a distance of 3 kilometer from one another, we merge those two clusters into one.

Flowchart:



Continued on next page

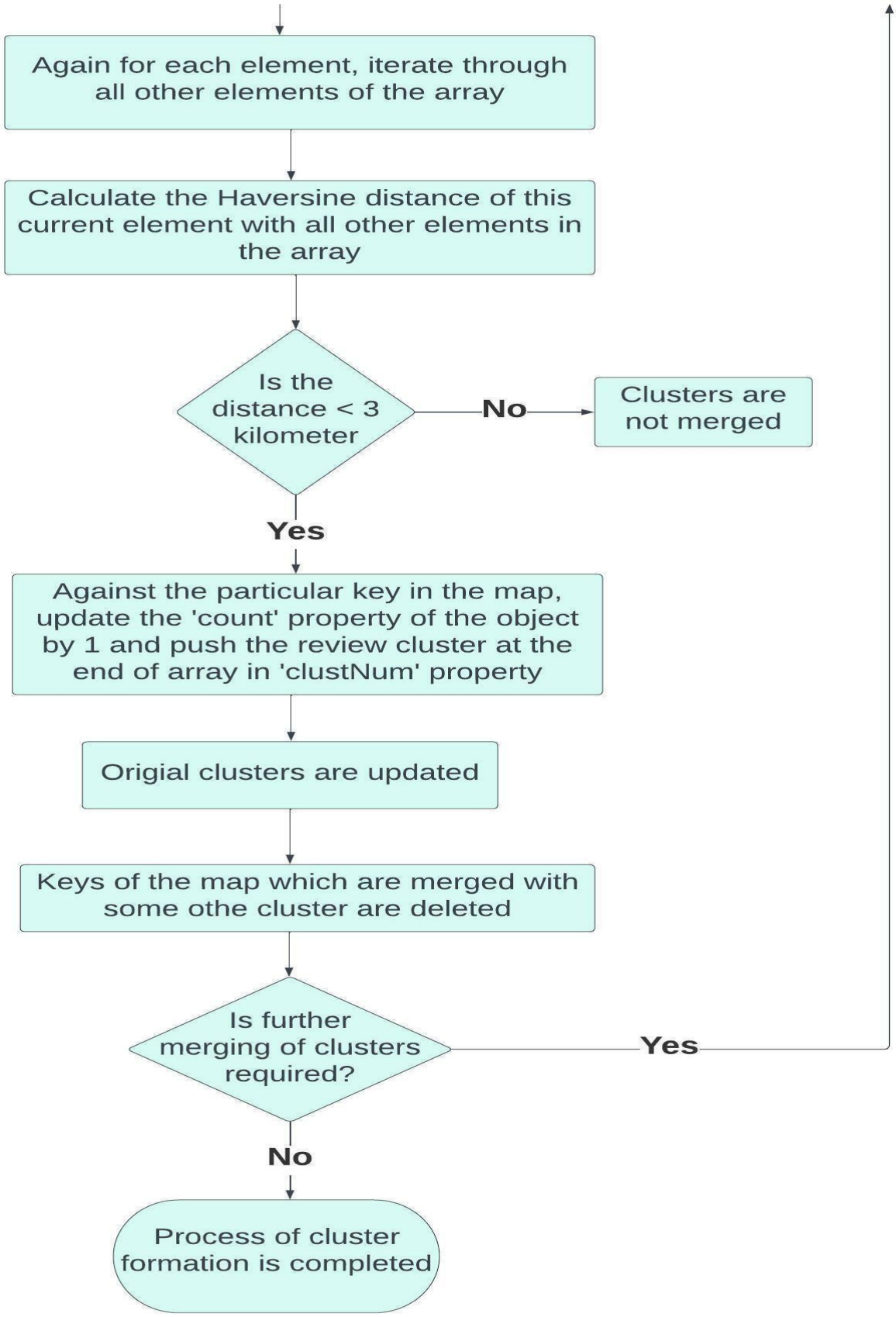


Fig 3: Final clusters after merging based on closeness of sources / distances of the routes reviewed

Steps (As discussed in Fig 3):

i) All the clusters that we have formed so far either based on same source or same destination, are now taken as input for this phase. These clusters are stored into an array of objects with the following properties-

i) type : it indicates whether the cluster is formed based on source or destination

ii) info : it holds all the routes' information inside a particular cluster

Code Snippet:

```
for(let i=0;i<=selects.upto;i++){

  clust.push({type:selects.select===0?"source":"destination",
                    info:iterateMap[iterateKey[i]]});
            iterateMap[iterateKey[i]].map((item)=>{
                removeList[item._id]=item;
            }) }
         tempReviewList=tempReviewList.filter((item)=>{
            return !removeList[item._id] })
```

ii) We create two empty map objects, one for source, another for destination and iterate the array which stores our clusters. We check whether a particular cluster has been formed based on source or destination which is indicated by the 'type' property of an individual cluster.

If it has been based on source, we create a key in the source map against the source of this particular cluster and store an object against it with the following properties:

a) cluster : It stores the index at which the cluster is present in the original array in which all the clusters from the previous step has been stored. This information will help us to decide which are the clusters that should be merged later.

b) latlng: It's an array containing the latitude-longitude points of the source

If the cluster has been formed on the basis of destination, we create a key in the destination map instead against the destination of this particular cluster and store an object against it with the same two properties

a) cluster : It stores the index at which the cluster is present in the original array in which all the clusters from the previous step has been stored.

b) latlng : It's an array containing the latitude-longitude points of the destination.

Code Snippet for the above logic is as follows:

```
Const                    source={},dest={};
     clust.map((item,index)=>{
         if(item.type==="source"){
             source[item.info[0].from]={
                 cluster:index,

   latlng:[item.info[0].pathLat[0],item.info[0].pathLong[0]]
             }; }
         else{
             const  l1=item.info[0].pathLat.length;
             const l2=item.info[0].pathLong.length;
             dest[item.info[0].to]={
                 cluster:index,
                 latlng:[item.info[0].pathLat[l1-
   1],item.info[0].pathLong[l2-1]] }; } })
```

iii) Now, we extract the keys, i.e. sources and destinations from both the maps and store it into two separate arrays. One array containing all the sources and the other one containing all the destinations.

Code Snippet:

```
let sourceKey=Object.keys(source),destKey=Object.keys(dest);
```

iv) We set the value of 'Distance' parameter to be 3. It means that any pair of sources or destinations are considered to be close to one another if they are within a distance of 3 km.

v) We iterate the array obtained in step iii), which contains all the unique sources and for every individual source, we compare its distance with all other distinct sources in the array.

If the distance between a pair of sources is within our set threshold value (i.e. 3 km), we again create a map with the initial source name that we have started comparing with as a key and store an object against it with the following properties:

count: It indicates how many of those other distinct sources are within 3 km distance from this particular source.

clusternum: It's an array which contains the index number of those sources that are within 3 km distance from this particular source. This index number will help us to track which are the clusters that should be merged together.

After all the elements in the source array have been iterated, if we take a look at our map that we have created in this step, for any particular source as a key on the map, its '***count***' property will tell us how many other sources are within 3 km distance

from this particular source and its '***clusternum***' property will contain an array which contains the index number of all those sources and it will eventually help us to keep track while merging these clusters together.

Code Snippet for the above logic is as follows:

```
const sourceMap={};
       sourceKey.map((item)=>{
           sourceMap[item]={count:0,
           clusterNum:[]};
           sourceKey.map((itm)=>{
                   if(item!==itm &&
distance(source[item].latlng[0],

source[item].latlng[1],source[itm].latlng[0],
                    source[itm].latlng[1]) < D) {

                    sourceMap[item].count++;
                    sourceMap[item].clusterNum.
                   push(source[itm].cluster); } }) })
```

vi) In this step, we iterate the destination array obtained in step iii) and do the exact same things that we have done in step v).

After step v) and vi), the clusters that should be merged with one another are identified.

vii) If two clusters need to be merged together, we decide whether the first cluster should be merged with the second one or vice versa.

The logic behind merging of clusters is as follows:

If there are n number of clusters that should be merged together, we check the value of 'count' property of each of these clusters. It indicates how many clusters should be merged with this particular cluster. So, the cluster which has highest 'count' value out of these n clusters should be the base cluster and the remaining n-1 clusters should be merged with this one.

Code Snippet:

```
function maxClust(arrKey,Map){
         let max=Map[arrKey[0]].count;
         let        element=arrKey[0];
         arrKey.map((item)=>{
         if(max<Map[item].count){
             element=item;
             max=Map[item].count;
         } })
        return element; }
```

```
const                          element=maxClust(sourceKey,sourceMap);
        newCluster[`C${source[element].cluster}`]={
            type:str,
            info:clust[source[element].cluster].info };
        sourceMap[element].clusterNum.map((item)=>{
            newCluster[`C${source[element].cluster}`].info
            =newCluster[`C${source[element].cluster}`].info
            .concat(clust[item].info);
            delete source[clust[item].info[0].from]; })
        delete                          source[element];
        sourceKey=Object.keys(source);
```

In this way we ensure that maximum number of clusters are merged together as possible, thereby increasing the cluster size as much as possible which in turn will increase the areas / routes covered by a particular cluster as much as possible. Bigger the cluster size, wider the areas covered by a particular cluster, the need of implementing privacy will be more.

Now, we have completed the process of cluster formation. Depending upon whether a cluster is based on source or destination, all the routes inside any particular cluster will either have same source (or, destination) or the sources (or, destinations) are within a distance of 3 kilometer.

Fig 4: Clusters that have been formed on the basis of all the routes reviewed by a user

In Fig 4, we can see that each cluster contains the type (either source or destination) based on which it has been formed. It also contains the number of routes that are present inside it.

If we click on the, 'view me' option for any cluster, it will take us to another page where all the routes present inside that cluster will be displayed.

For example, we want to check all the routes that are there inside ‘Cluster1’.

Back

From	To	Path Latitudes	Path Longitudes	Reviews(in order)	Created At
Jadavpur Police Station Bus Stop, Raja Subodh Chandra Mullick Road, Poddar Nagar, Kolkata 700032, West Bengal	Kulpi Baruipur Road, Khodar Bazar Uttarpara, Baruipur 700144, West Bengal	[Array]	[Array]	[5,5,5,5,5]	2022-03-01T19:18:27.546Z
Jadavpur Sukanta Market, B/140, Raja Subodh Chandra Mullick Road, KPC Medical College & Hospital Campus, Kolkata 700032, West Bengal	Naktala Metro Station, Adi Ganga Road, Bidhan Pally, Kolkata 700084, West Bengal	[Array]	[Array]	[5,5,4,3,5]	2022-02-21T15:33:19.718Z
Bawarchi Family Restro, 44, Raja Subodh Chandra Mullick Road, Baghajatin, Kolkata 700092, West Bengal	Survey Park Pathagar, Santoshpur Avenue, Purba Diganta, Kolkata 700075, West Bengal	[Array]	[Array]	[5,5,2,2,3]	2022-02-26T17:59:21.017Z
Jadavpur Thana More, Prince Anwar Shah Road, Jadavpur University Campus, Kolkata 700032, West Bengal	Jadavpur Railway Station, Doctor Kiran Sankar Roy Road, KPC Medical College & Hospital Campus, Kolkata 700032, West Bengal	[Array]	[Array]	[5,5,1,5,1]	2022-02-26T18:02:32.228Z
8, Jadavpur Central Road, Poddar Nagar, Kolkata 700032, West Bengal	Baghajatin Park, Kolkata, West Bengal	[Array]	[Array]	[4,5,4,2,4]	2022-02-21T15:30:52.613Z
Tollygunge Metro Station Bus Stop, Deshpran Sashmal Road, Tollygunge Golf Club, Kolkata 700040, West Bengal	South City Mall, 375, Prince Anwar Shah Road, South City Complex, Kolkata 700029, West Bengal	[Array]	[Array]	[5,5,2,3,5]	2022-02-21T15:15:52.215Z

Figure 5: Routes inside ‘Cluster1’

Once the clusters are formed, by taking a look at the number of routes inside each cluster, we can easily say that whether a cluster contains significantly greater number of routes than other clusters. If yes, it indicates that the user frequently travels in those routes which belong to that particular cluster.

We then further apply our privacy metric on the clusters formed to reduce the size of cluster in order to prevent any adversary to infer about the whereabouts of our user by seeing the routes that s/he have reviewed.

5.2.5. Implementing Privacy Metric

After forming the clusters in section 5.2.4, we mainly perform two tasks in this section:

5.2.5.1 Identifying the routes inside a cluster that should be removed

We first decide which are the routes that can be removed from a particular cluster in order to reduce cluster size while also ensuring that the overall accuracy of the route information is very less compromised.

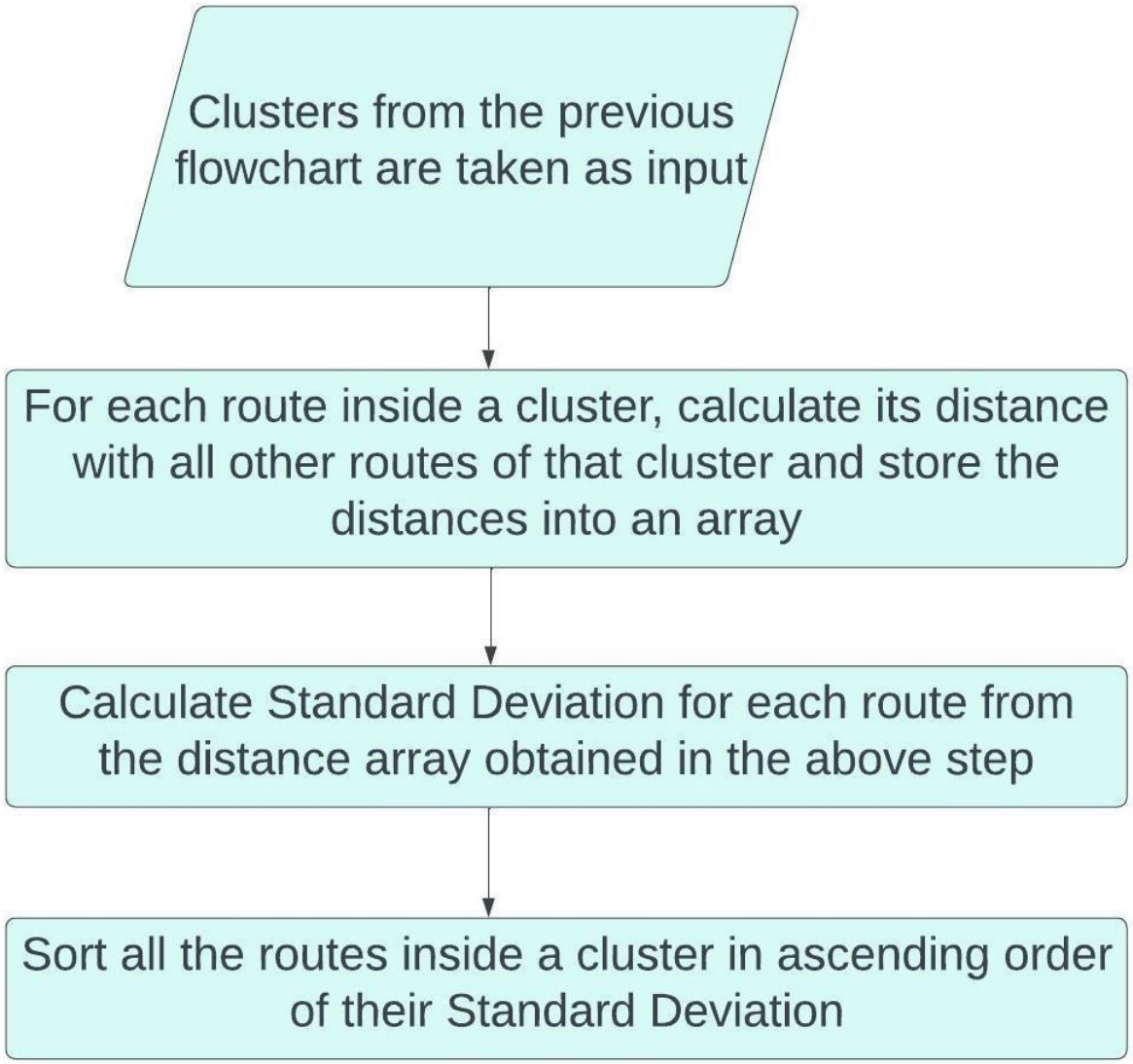


Fig 6: This flowchart determines on what basis routes should be removed from a cluster

We first calculate the distance between each pair of routes present inside a cluster. In order to calculate distance between any two routes, we simply traverse each point of the latitude-longitude arrays of those two routes, and for a pair of latitude-longitude points of any two routes we find out the distance between them using Haversine Distance formula. Thus, we

keep on calculating distance between all pairs of latitude-longitude points of one route with the corresponding latitude-longitude points of another route and keep on summing up these calculated distances. The final cumulative value is the actual distance between any two routes. Therefore, if there are n number of routes in total, for each route we will have an array of n-1 elements where each element indicates its distance with some other route which also belongs to the same cluster.

Now, we find out the Standard Deviation of these n-1 array elements and this deviation value is the final indicator whether a particular route should be removed first or last while removing routes from the cluster to which it belongs to.

Code Snippet for the above logic is as follows:

```
function rearrangeClusters(info,type){
        let             totalArray={};
        if(info.length==1){
            totalArray[info[0]['_id']]=1;
            return totalArray;
        }
        info.map((item,index)=>{
            for(let i=0;i<info.length;i++){
                if(i==index)
                    continue;
                let    sum=0;
                for(let
k=0;k<Math.min(item.pathLat.length,info[i].pathLat.length);k++){
                    const                val=k;
                    if(type=='destination')

k=Math.min(item.pathLat.length,info[i].pathLat.length)-1-k;
                    const
dist=distance(item.pathLat[k],item.pathLong[k],info[i].pathLat[k],in
fo[i].pathLong[k]);
                    sum+=dist;
                    k=val; }

                if(!totalArray[item['_id']])
                    totalArray[item['_id']]=[sum];
else
                    totalArray[item['_id']].push(sum);
 }

            const temp=totalArray[item['_id']];
            console.log(temp);
            const       numbers       =       new       DeviationStream();
            temp.map((item)=>numbers.push(item));
            totalArray[item['_id']]=numbers.standardDeviation() })
        console.log(totalArray);
        return totalArray; }
```

A lower value of deviation means that all other routes inside that cluster are within close proximity of this particular route and higher value of deviation for a particular route indicates that this particular route even though it belongs to the same cluster, still the areas

covered in that route are not within very close distance of all other routes which belong to the same cluster. So, we can say that while removing routes from a cluster, the route with lowest value of Standard Deviation should be removed first in order to make sure that the accuracy of data is less compromised.

5.2.5.2. Identifying the clusters which are significantly bigger in size than other clusters

We identify the clusters which contain significantly greater number of routes than the other ones. We then decide how many routes should be removed from such a cluster so that there exists no clear peek in the number of routes inside a cluster than the other clusters.

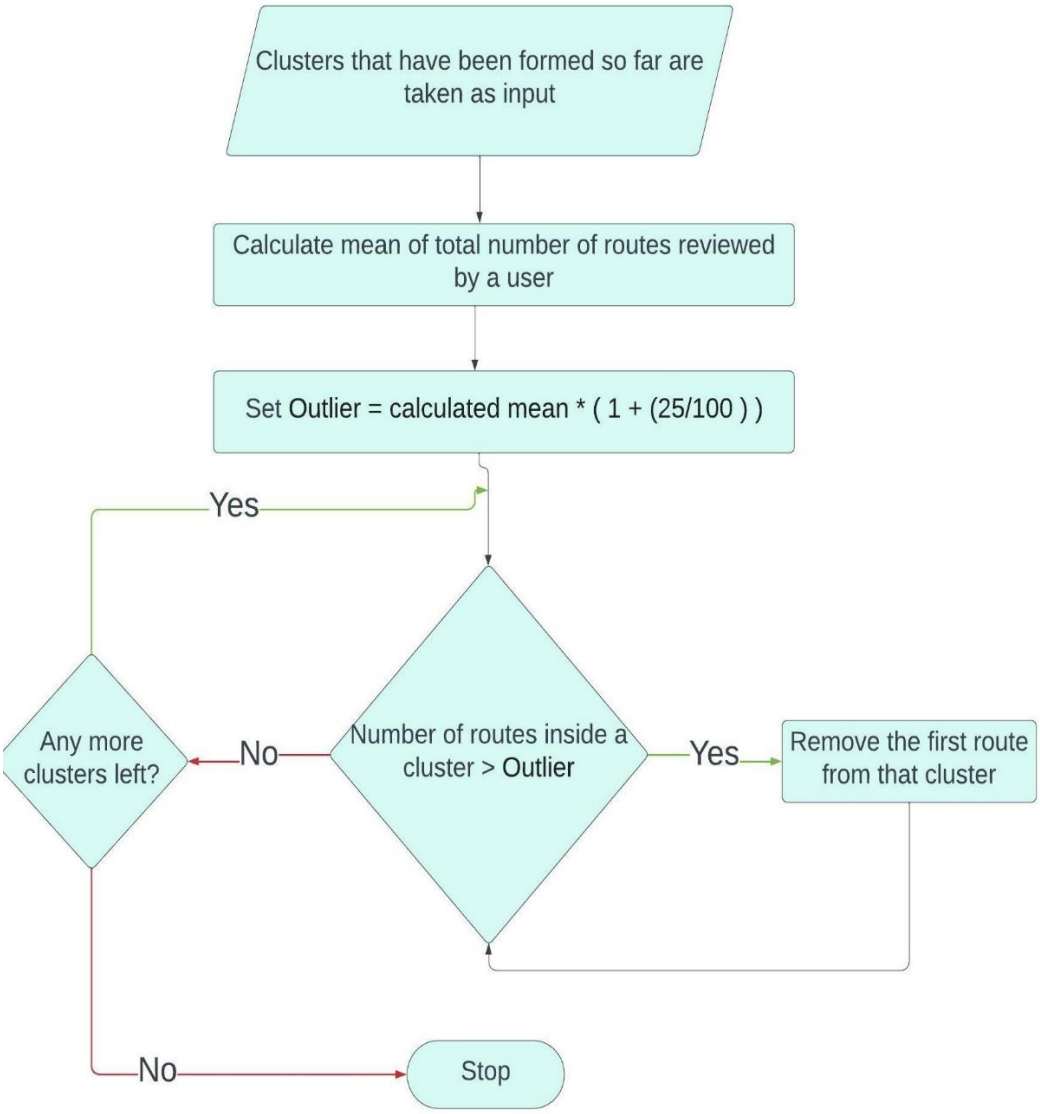


Fig 7: This flowchart shows how clusters significantly bigger than other ones are being identified and routes are being removed in order to reduce the size

This problem is similar to the problem of removing outliers from an array of n elements. Here, each element corresponds to one individual cluster size. We set a value of outlier and any cluster having more number of routes than this outlier value is identified and trimmed.

As illustrated in Fig 7, we take each cluster one by one and store the number of routes present inside that cluster in an array. Thus, after all the clusters have been taken into consideration, each array element gives us the number of routes that is present inside a particular cluster. For example, element at 0^{th} index gives us the number of routes in Cluster-0, element at 1^{st} index gives the number of routes in Cluster-1 and so on.

Now, we calculate the mean of these array elements. This gives us the mean of total number of routes (irrespective of which cluster a particular route belongs to) rated by a user.

Next, we calculate the value of outlier as follows:

Outlier = (mean + 0.25 * mean) [mean of total number of route reviews submitted by a user]

Any cluster which contains more number of routes than this calculated value of outlier, we take those clusters and start removing routes one by one from a cluster until the number of routes in that cluster becomes less than our calculated value of outlier.

Code Snippet for the above logic is as follows:

```
function reduceOutliers() {
        const epsilon=25;
        const numbers = new DeviationStream();
        let clustercpy=[...cluster];
        clustercpy.map((item)=>numbers.push(item.info.length));
        const mean=numbers.mean();
        clustercpy.map((item)=>{
            while(item.info.length>mean*(1+(epsilon/100)))
                item.info.shift(); })
        setCluster(clustercpy) }
```

Once this process is completed, we can say that our dataset is now differentially private.

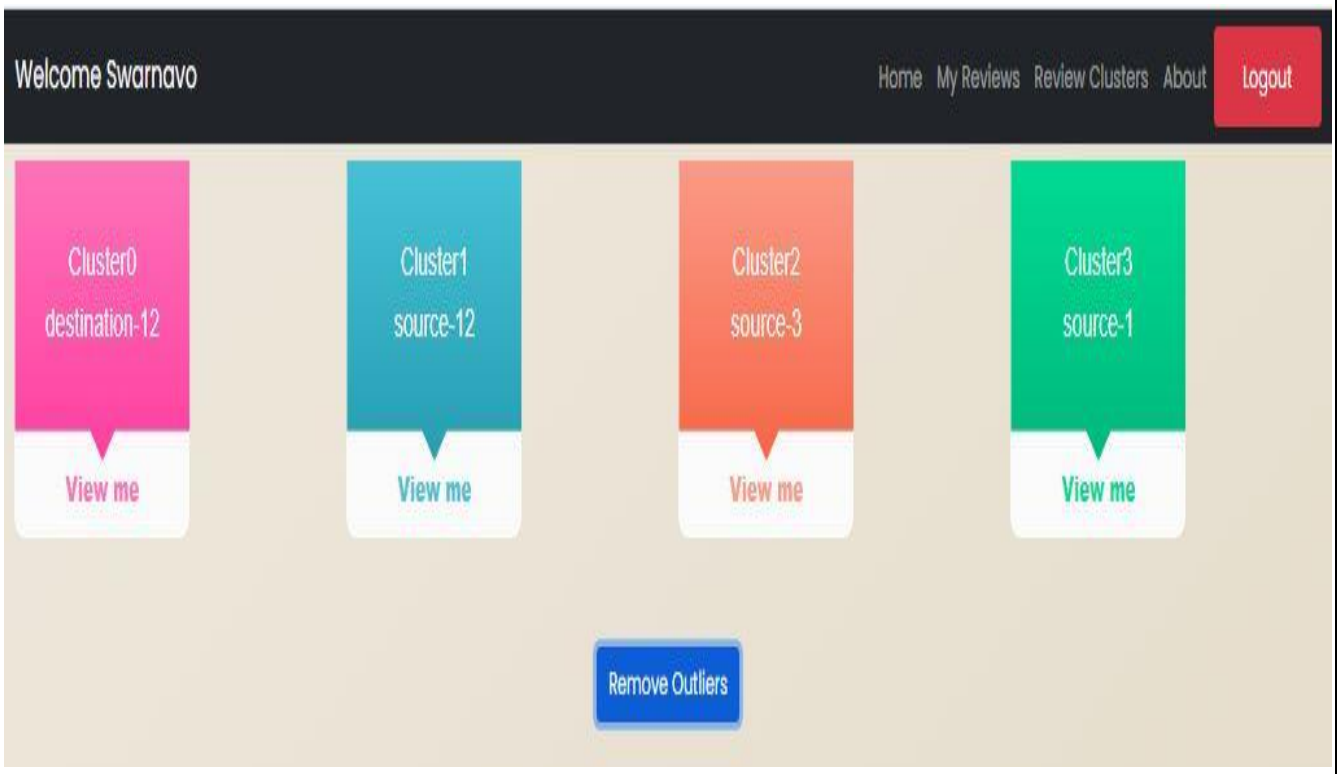


Fig 8: Reduction in cluster size after introducing noise in the dataset

Chapter 6: Experimental Results

In this section, we will analyze how an adversary may infer about the whereabouts of a user once the dataset containing all the routes that any user has reviewed is made public. In this context, we will further see how and to what extent noise introduced by Differential Privacy in the dataset can help to prevent such inference attacks.

6.1. Analysis of User Reviews before and after implementing Differential privacy

Here, we will take a number of different users and analyze how changes in user behaviour i.e. the number of routes reviewed by them, areas covered in the routes reviewed, determines the level of required privacy and corresponding data loss in order to make their review dataset differentially private.

User1:

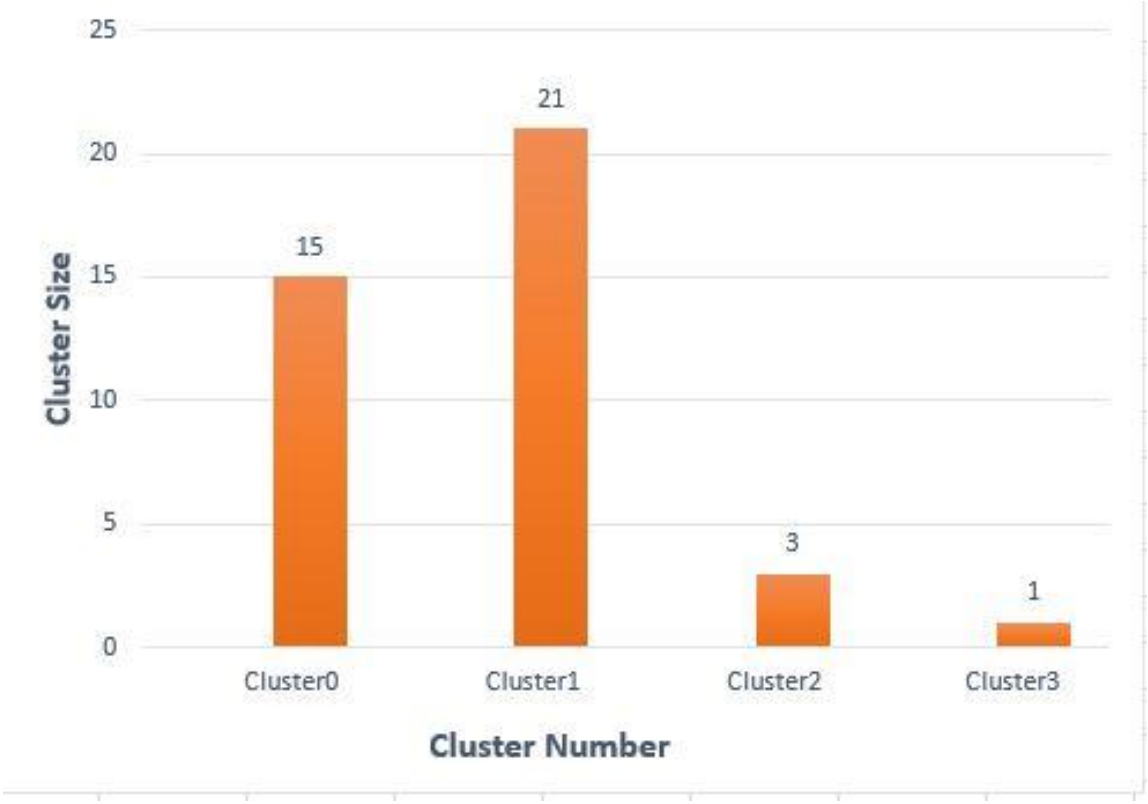


Fig 9: Cluster Size before implementing privacy metric

As we can see in Fig 9, this particular user has reviewed a total of (15+21+3+1) = 40 routes out of which 21 routes fall under the same cluster and another 15 routes fall under a single cluster. This clear peak in the size of these two clusters indicates that the user frequently travels in the areas covered under these clusters. This inference can be risky in context of the location privacy of user. So, we further introduce our privacy metric to reduce the cluster size.

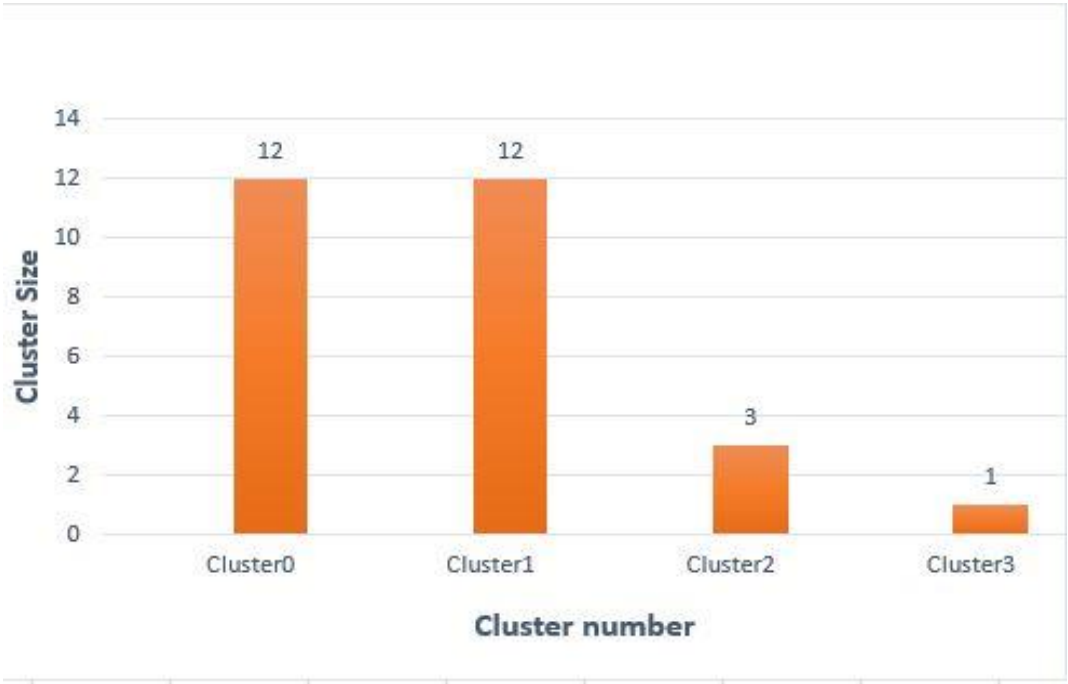


Fig 10: Cluster Size after implementing privacy metric

Fig 10 shows how we have been able to control the odd peak in size of 'Cluster0' and 'Cluster1' as shown in Fig 9 by introducing noise in the dataset.

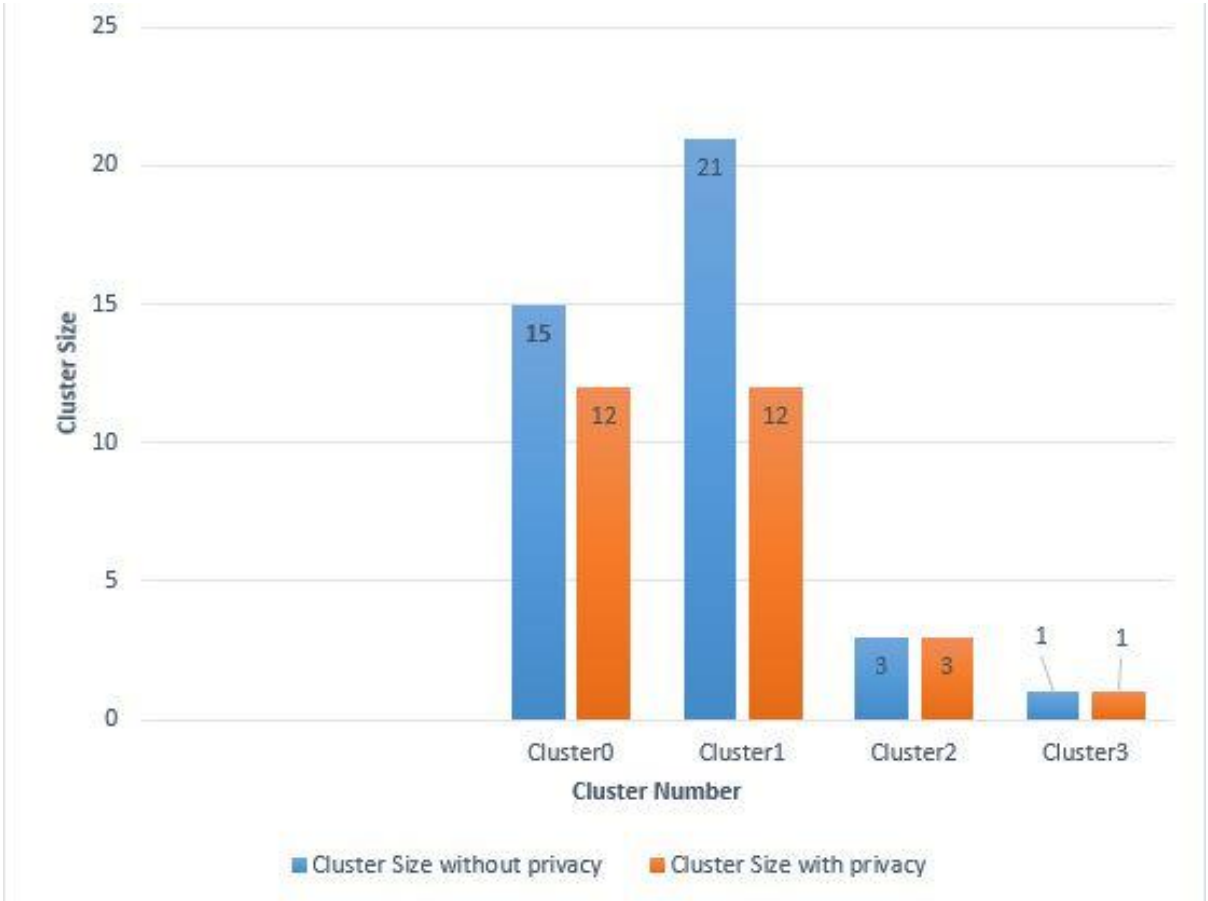


Fig 11: Comparison of cluster size before and after implementing privacy metric

Fig 11 shows how the size of 'Cluster0' and 'Cluster1' has been reduced from 15 to 12 and 21 to 12 respectively as an effect of adding noise in the dataset. Size of other three clusters remain unchanged. However, the ratio of cluster size before and after implementing privacy metric is not changed much. The clusters which were bigger before implementing privacy will still be bigger in size after implementing privacy metric. This shows that the overall accuracy of the information is not hampered much as an adverse effect of implementing privacy.

User2:

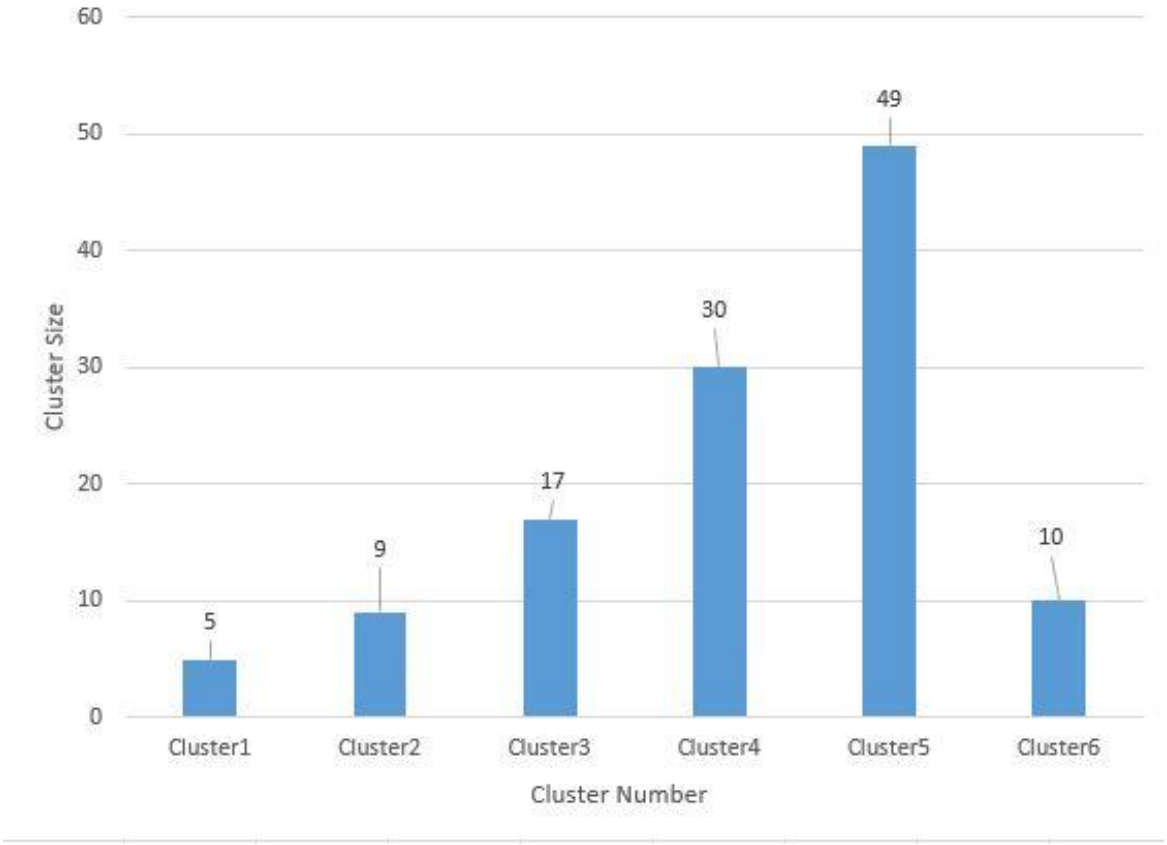


Fig 12: Initial Cluster Size before implementing privacy metric (User2)

Fig 12 shows that this particular user has reviewed a total of (5+9+17+30+49+10) = 120 routes and depending upon the areas covered by those routes, six clusters have been formed. Out of these six clusters, 'Cluster4' and 'Cluster5' exhibit odd peak in their cluster size which might help any adversary to infer that this particular user mostly travels in those routes/areas covered in 'Cluster4' and 'Cluster5'.

We will now see how such risk of inference attack can be prevented by adding noise in the original dataset.

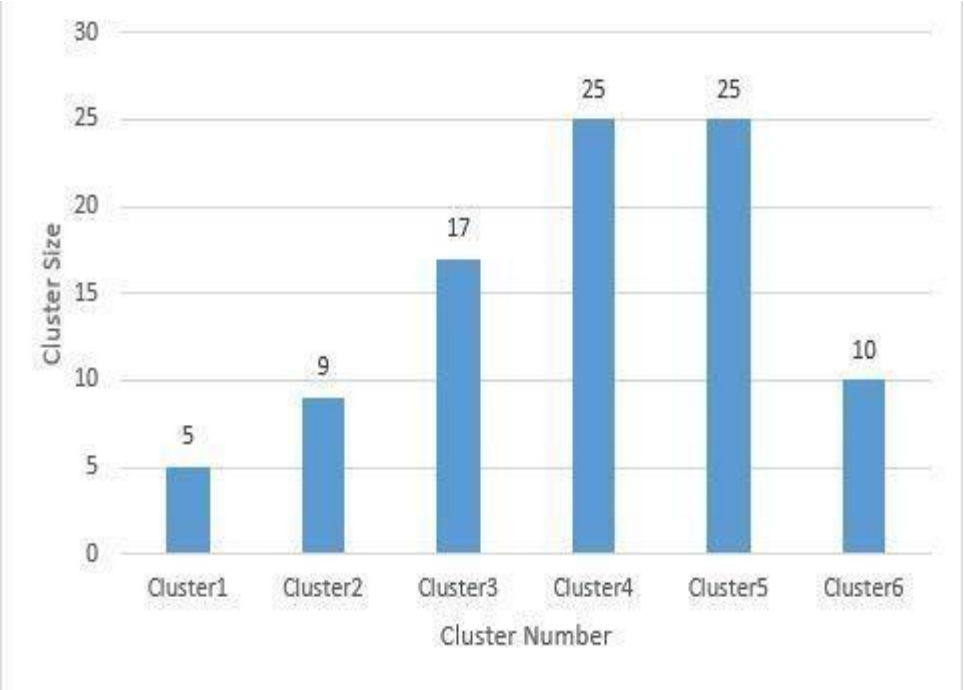


Fig 13: Change in cluster size after implementing privacy metric (User2)

Fig 13 helps us understand how the clusters having significantly greater number of reviews are being trimmed in order to remove any obvious peak in any of the cluster size. 'Cluster4' and 'Cluster5' which displayed obvious peak in their size in Fig 12, have now been trimmed and in harmony with the size of other clusters.

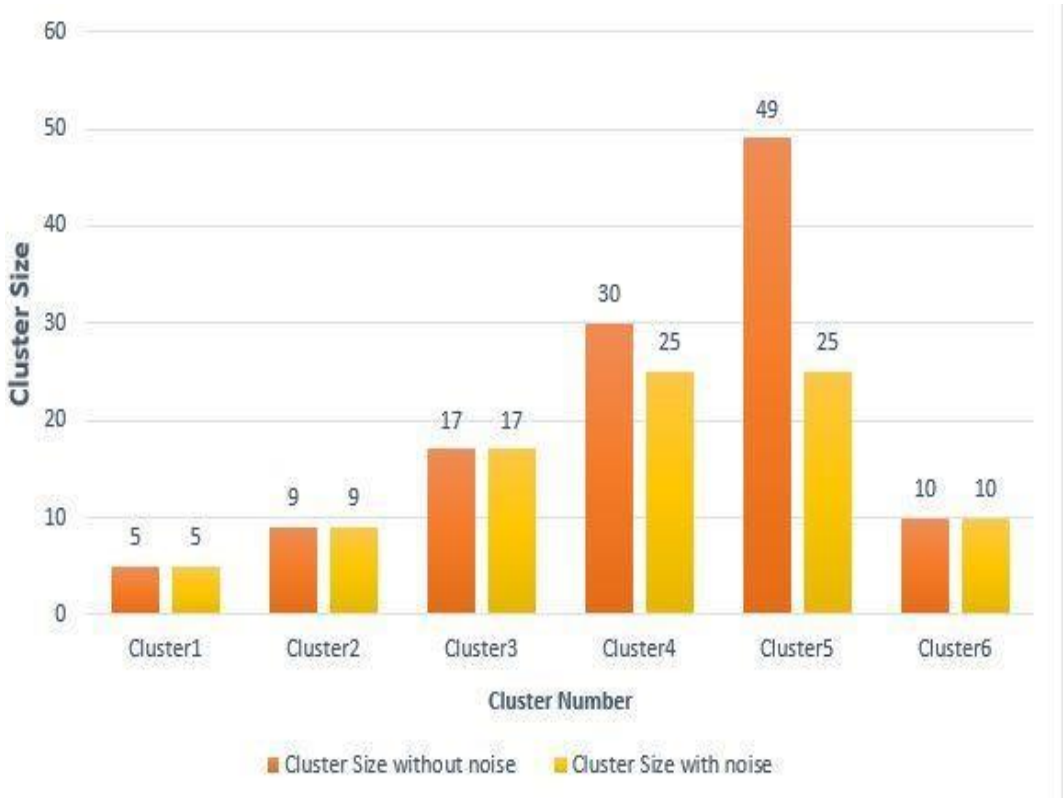


Fig 14: Comparison of cluster size before and after implementing privacy metric (User2)

As we can observe from Fig 14, size of 'Cluster4' and 'Cluster5' have been reduced by 5 and 24 reviews respectively after adding noise, while the size of other clusters remains same.

As discussed w.r.t Fig 11, here also, the size ratio of the clusters is not changed much before and after implementing privacy metric. The clusters which contained more routes before implementing privacy will still contain higher number of routes than other clusters after implementing privacy. It shows that the overall meaning and accuracy of user information is not hampered much.

User3:

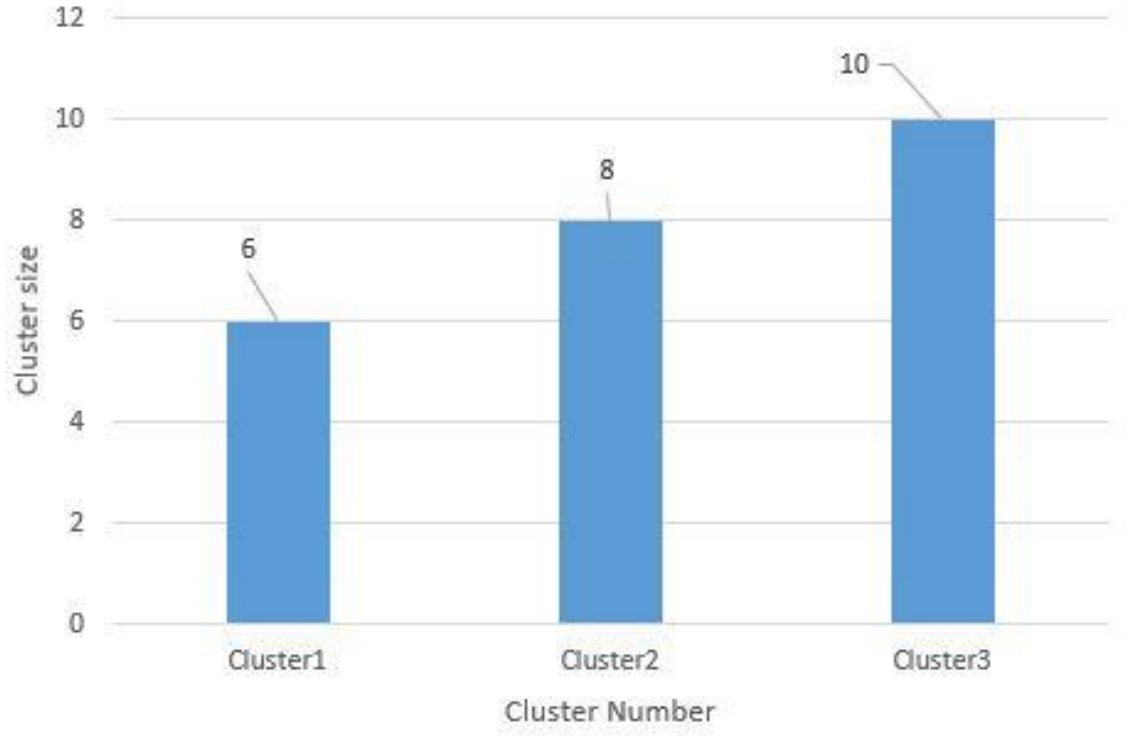


Fig 15: Size of clusters before implementing privacy metric (User3)

As we can see in Fig 15, this user has reviewed a total of (6+8+10) = 24 routes. However, the differences in cluster size is very less for all the clusters. There is no cluster with significantly bigger size than other ones.

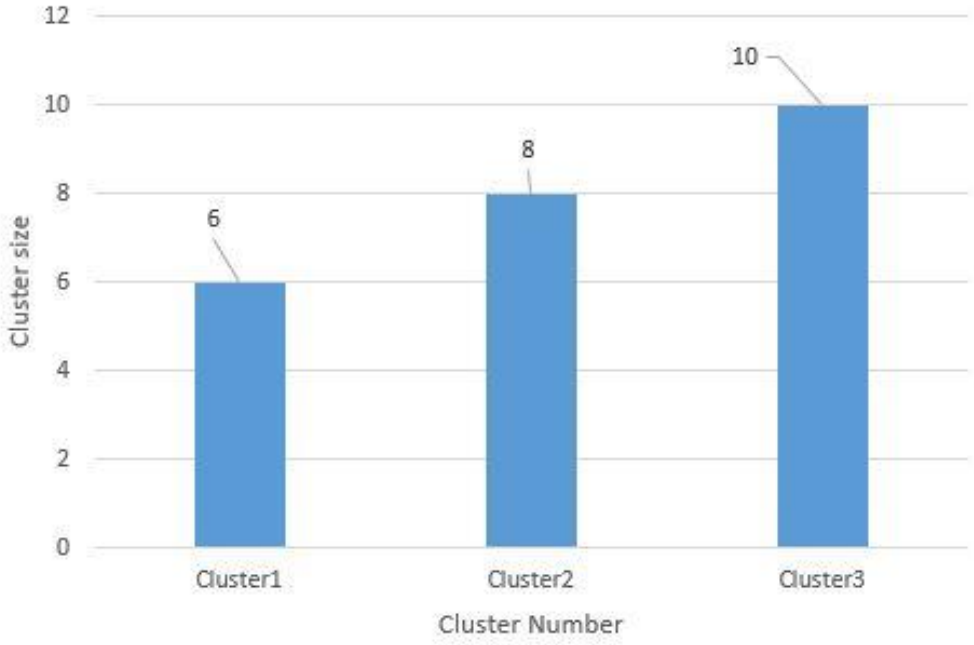


Fig 16: Size of clusters after implementing privacy metric (User3)

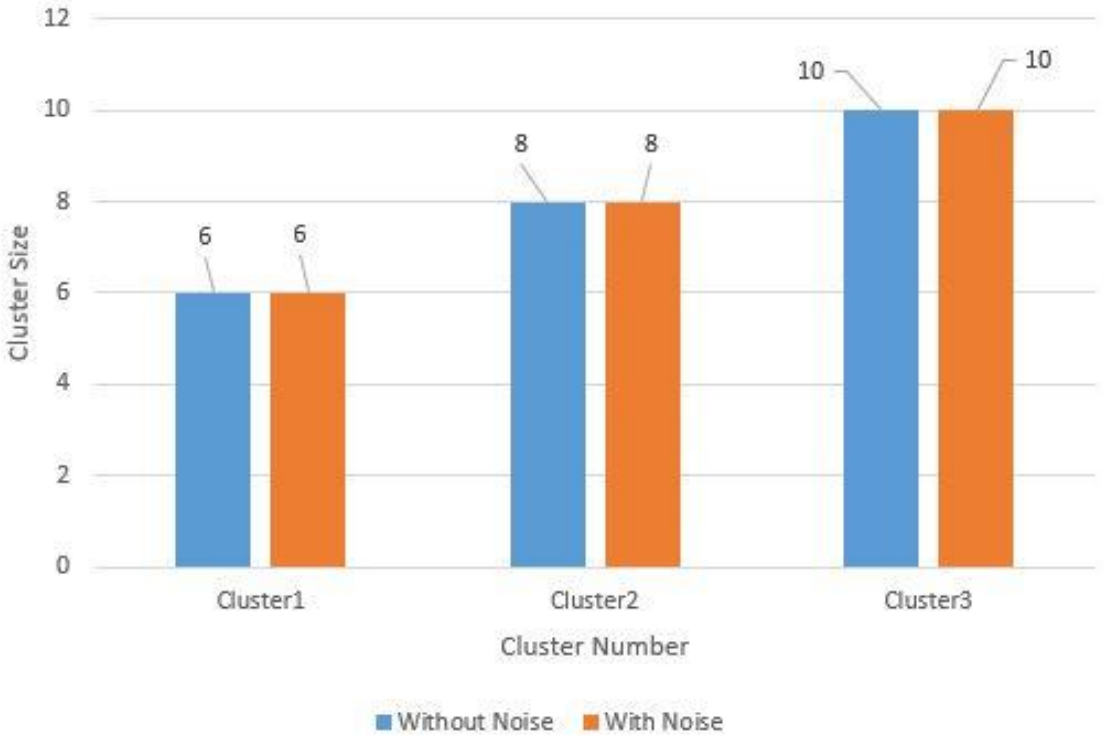


Fig 17: Cluster size before and after implementing privacy metric

Fig 17 gives us a very interesting result. As we can see that there is no change in any of the cluster size for this particular user before and after introducing noise in the dataset. This is because the size of original clusters is very close to each other. It proves that the level of noise to be introduced in the dataset depends upon the variation in size of the clusters formed.

	Cluster Distributions		Standard Deviation	
	Without Privacy	With Privacy	Without Privacy	With Privacy
User1	15,21,3,1	12,12,3,1	9.59	5.83
User2	5,9,17,30,49,10	5,9,17,25,25,10	16.7	8.54
User3	6,8,10	6,8,10	2	2
User4	6,8,11,15	6,8,11,12	3.92	2.75
User5	1,4,12,20,22,26,30,35,45	1,4,12,20,22,26,27,27,27	14.34	10.28

Fig 18: Table of Standard Deviation of different distributions of our dataset before and after implementing privacy metric

Fig 18 shows the Standard Deviation values of different distributions of our dataset of different users before and after implementing privacy metric.

	$Z\text{-Score}_{real\ data}$	P(in)	$Z\text{-Score}_{pertubed\ data}$	P(out)	ε
User1	1.14	0.87286	0.85	0.80234	0.08424
User2	1.74	0.95907	1.15	0.87493	0.09182
User3	1	0.84134	1	0.84134	0
User4	1.27	0.89796	1	0.84134	0.06513
User5	1.63	0.94845	0.83	0.79673	0.17431

Fig 19: Table of calculated value of epsilon for dataset of different users

As we can see in Fig 19, we get the value of P(in) from Z-Score of real data and P(out) from Z-Score of perturbed data. The process of calculating Z-Score and epsilon is mentioned before in section 4.2.4.

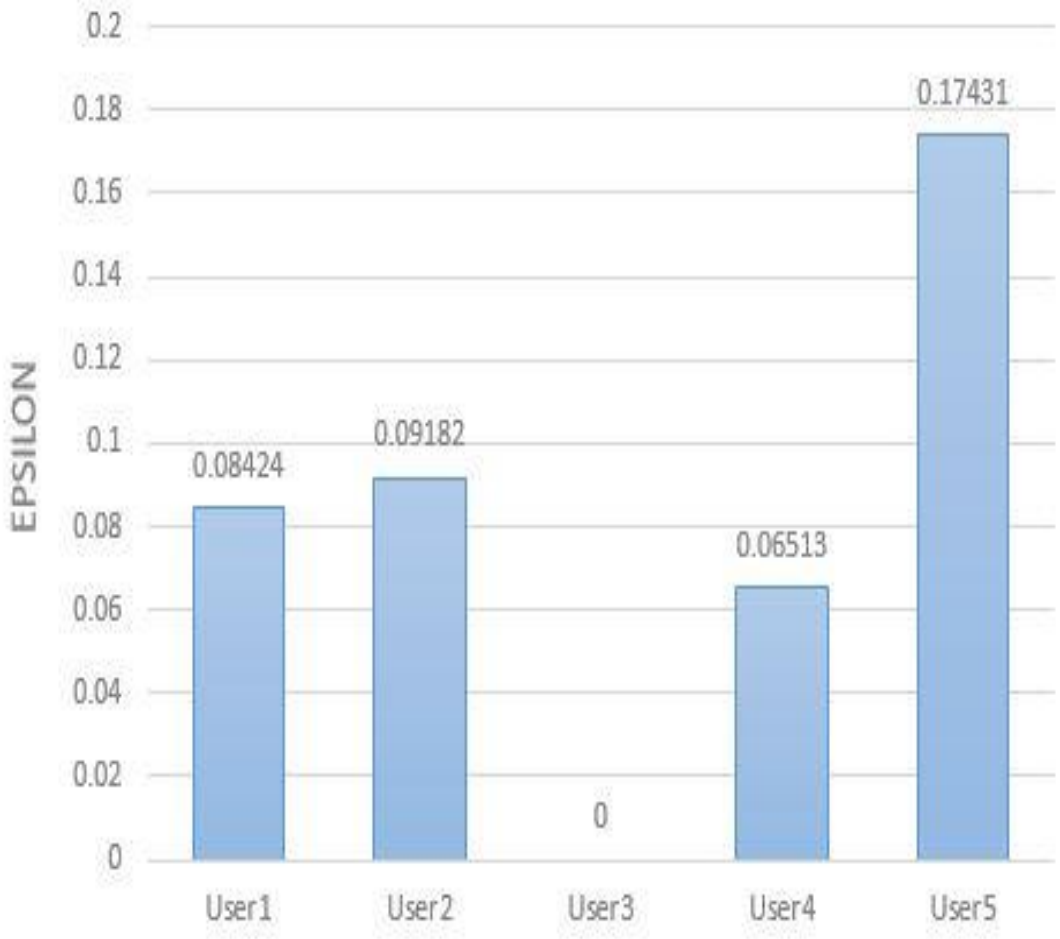


Fig 20: Value of epsilon calculated for different user's dataset.

Fig 20 gives us the value of epsilon for datasets of different users. Higher the value of epsilon, more private the data is.

Chapter 7: Conclusion and Future Scope

In this project, the challenge of preserving privacy of users in a Crowdsourced Road Monitoring Application has been dealt with. The dataset of any user contains information about the roads/areas in which they travel. This information can become sensitive for the users who frequently travels in same roads/areas. Seeing this road information of any user, any adversary might be able to identify the roads/areas in which the user travels more frequently and thereby infer about the user's residential location or workplace. It possesses great safety and security concern of the users. However, the nature of the application demands these datasets of the users to be made public. Therefore, the solution is to implement Differential Privacy which allows a dataset to be made public only after adding some carefully calculated noise to the dataset in such a way that information about any individual can't be identified or predicted from that noisy dataset while also taking care of the accuracy of information. In our case, the first task was to identify those roads/areas from a user's dataset which are prone to inference attack. Next, the value of epsilon, i.e. the level of noise to be added in the dataset is carefully calculated considering the necessary parameters. Lastly, the calculated noise is added with the original dataset in order to make it differentially private and thereby eligible to be made public.

Future Scope:

i) Since this is a crowdsourced application, there is always a reliability concern about the correctness of data. The information that are being collected from users need to be validated first, before using it for further computation. So, in future, we plan to apply data validation techniques on the collected data. Multi phased data validation would be beneficial.

ii) Data collection from a sizeable population is challenging. So, we plan to involve more users and even perform a data collection campaign.

iii) We plan to design benchmark metrics to indicate the performance of the crowdsourced route recommendation systems that covers different perspectives.

References:

1. https://en.wikipedia.org/wiki/Crowdsourcing

2. https://sproutsocial.com/glossary/crowdsourcing

3. Howe, Jeff (2 June 2006). "Crowdsourcing: A Definition". Crowdsourcing Blog. Retrieved 2 January 2013.

4."Daren C. Brabham". USC Annenberg. University of Southern California. Retrieved 17 September 2014.

5. Brabham, Daren (2008), "Crowdsourcing as a Model for Problem Solving: An Introduction and Cases" (PDF), Convergence: The International Journal of Research into New Media Technologies, 14 (1): 75–90,

6. Guth, Kristen L.; Brabham, Daren C. (4 August 2017). "Finding the diamond in the rough: Exploring communication and platform in crowdsourcing performance". Communication Monographs. 84 (4): 510–533.

7. Estellés-Arolas, Enrique; González-Ladrón-de-Guevara, Fernando (2012), "Towards an Integrated Crowdsourcing Definition" (PDF), Journal of Information Science, 38 (2): 189–200,

8. Doan, A.; Ramarkrishnan, R.; Halevy, A. (2011), "Crowdsourcing Systems on the World Wide Web" (PDF), Communications of the ACM, 54 (4): 86–96,

9. Brabham, Daren C. (2013), Crowdsourcing, MIT Press, p. 45

10. https://www.skipsolabs.com/en/custom/news/view

11. Martin Abadi et al. “Deep learning with differential privacy”. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016, pp. 308–318.

12. Yang-Taek Oh and Jong Wook Kim. “Privacy-Preserving Traffic Volume Estimation by Leveraging Local Differential Privacy”. In: Journal of the Korea Society of Computer and Information 26.12 (2021), pp. 19–27.

13. Xiaokuan Zhang et al. “Defeating traffic analysis via differential privacy: a case study on streaming traffic”. In: International Journal of Information Security 21.3 (2022), pp. 689–706

14. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: 2010 ACM SIGMOD International Conference on Management of data. ACM (2010)

15. Chan, T.H.H., Shi, E., Song, D.: Private and continual release of statistics. ACM Trans. Inf. Syst. Secur. (TISSEC) 14, 1–24 (2011)

16. Jong Wook Kim et al. “Privacy-preserving data collection scheme on smartwatch platform”. In: 2019 IEEE International Conference on Consumer Electronics (ICCE). IEEE. 2019, pp. 1–4.

17. Abdur R Shahid and Sajedul Talukder. “Evaluating Machine Learning Models for Handwriting Recognition-based Systems under Local Differential Privacy”. In: 2021 Innovations in Intelligent Systems and Applications Conference (ASYU). IEEE. 2021, pp. 1–6.

18. https://en.wikipedia.org/wiki/Differential_privacy

19. Zheng H, Hu H, Han Z (2020) Preserving user privacy for machine learning: Local differential privacy or federated machine learning? IEEE Intelligent Systems 35(4):5–14