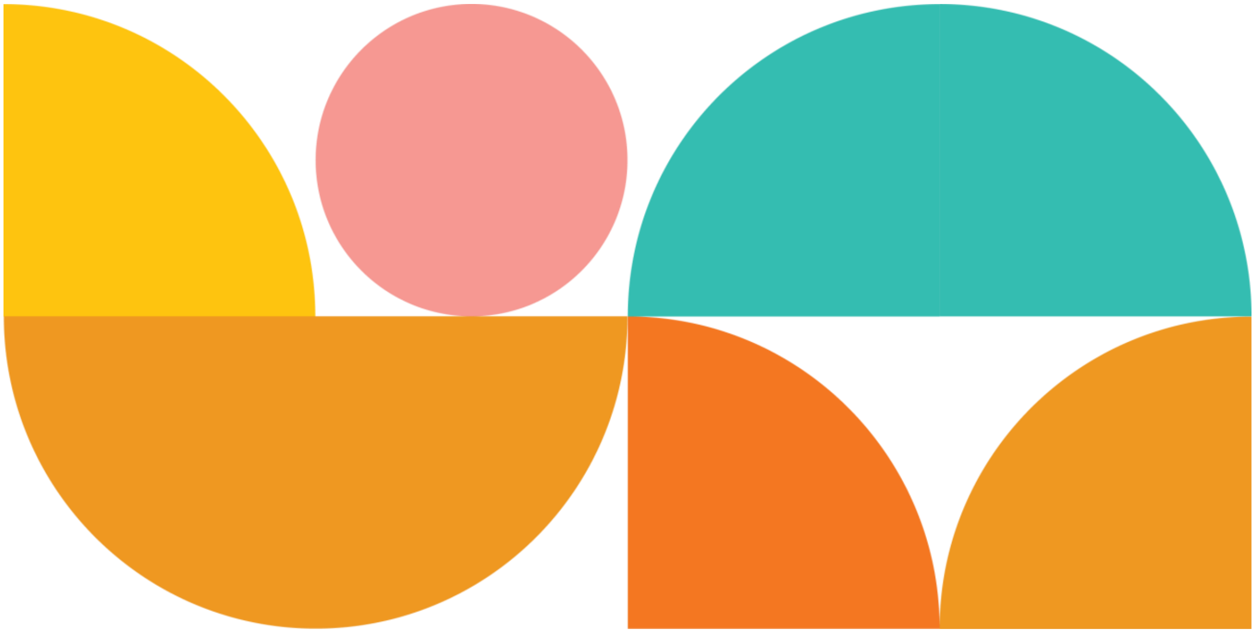
Contact tracing

B.Jignesh : 2201056



Problem Statement:

The COVID-19 pandemic highlighted the need for efficient and accurate contact tracing to prevent the rapid spread of the virus. Contact tracing involves identifying individuals who may have been exposed to the virus by determining their proximity to confirmed cases. However, traditional methods of contact tracing are time-consuming, prone to human error, and lack scalability.

This project aims to develop a machine learning-based contact tracing system to:

1. *Identify exposed individuals: Analyze geolocation data, proximity metrics, and interaction history to determine individuals who were in close contact with confirmed COVID-19 cases.*
2. *Alert exposed individuals: Provide timely notifications to those at risk and provide the data of nearby exposed people.*
3. *Visualize affected zones: Highlight areas with a high density of COVID-19 cases to inform individuals of nearby risks.*

The goal is to leverage data-driven insights to improve the speed and accuracy of contact tracing, reduce the spread of infection, and assist public health authorities in managing outbreaks effectively while ensuring user privacy and data security.

What is Contact Tracing?

- Contact tracing is the process of finding out who has recently been in close contact with a virus.

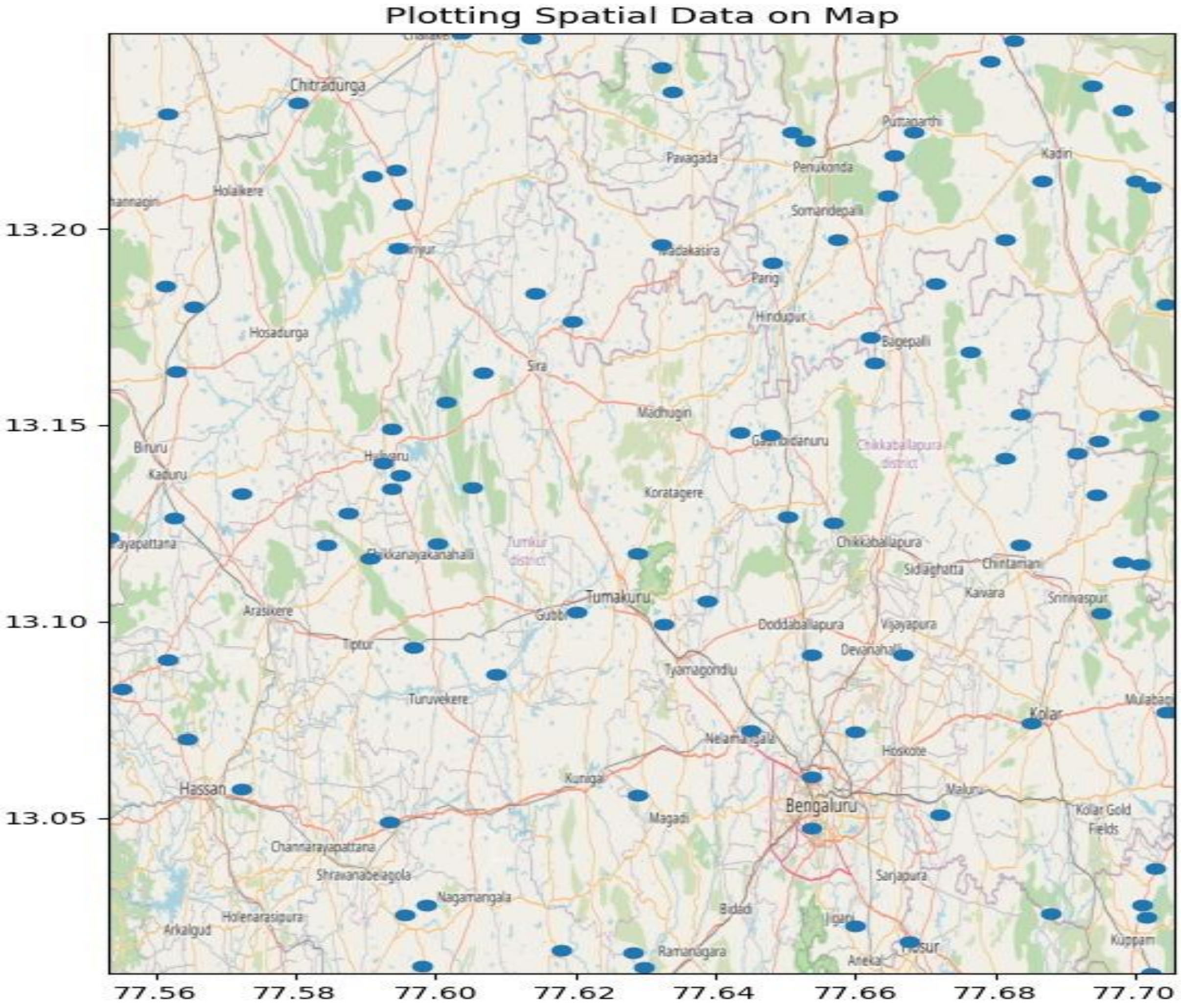
- Here in our context virus is COVID-19 and reaching out to those people to let them know, they may have been exposed.
- We also provide information about the nearby people who have been affected.

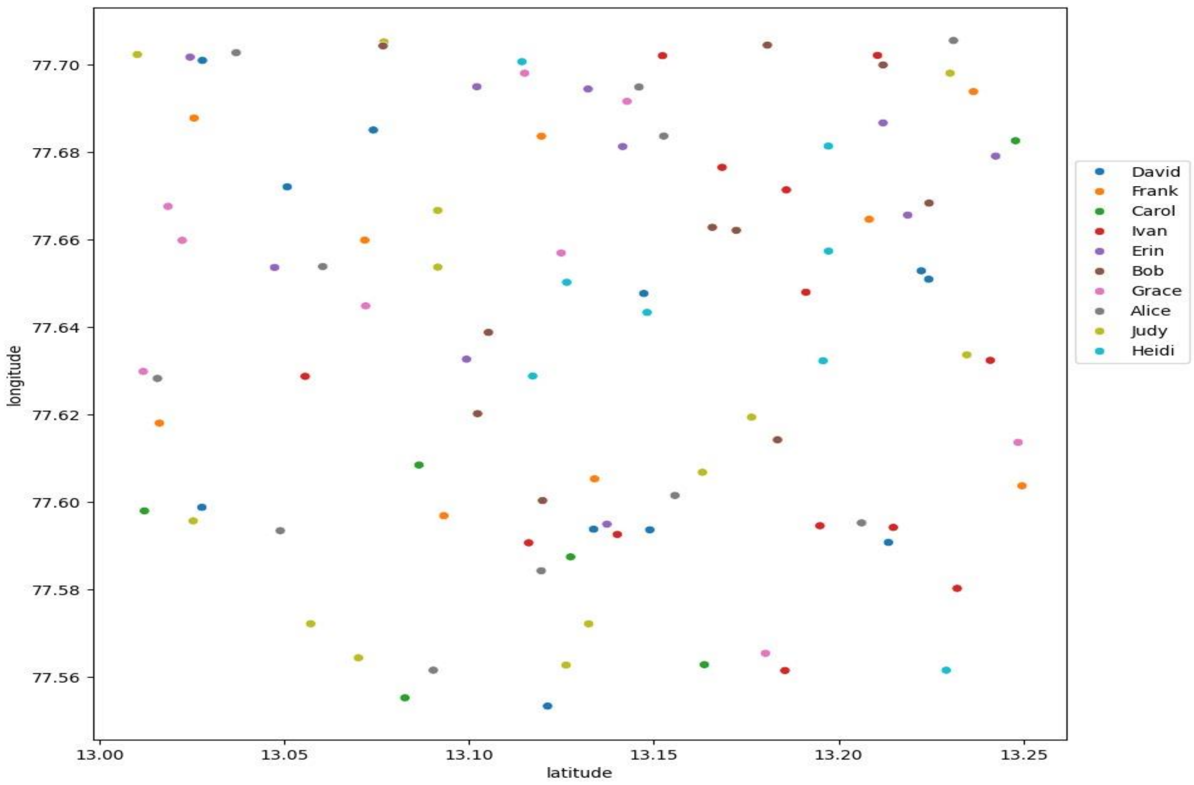
Which dataset?

- Took a real time dataset from Kaggle where it is data collected from Bangalore

- Dataset contains attributes id,latitude,longitude and this tuple defines the geographic location of particular person.
- There might be repetition of id’s in the dataset but with different latitude and longitude because that person might have travelled more than one place.

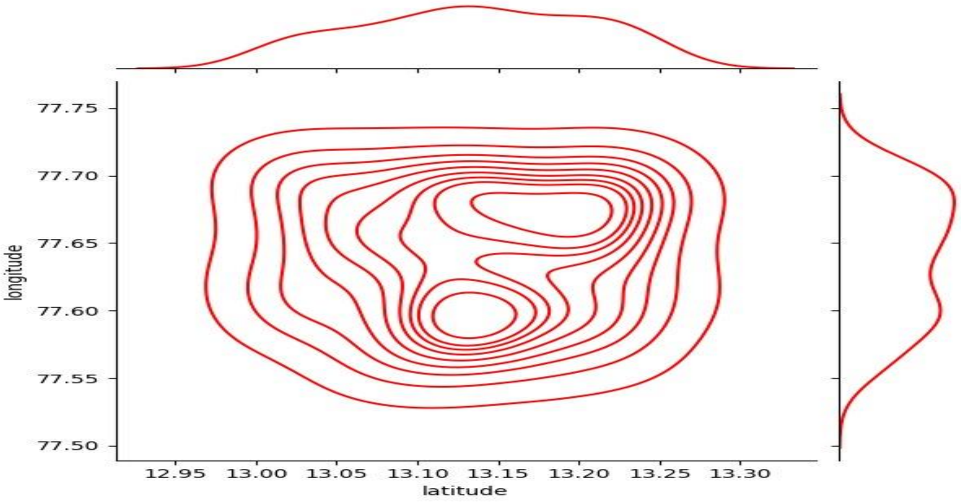
This is the scatter plot of latitude vs longitude for the data.



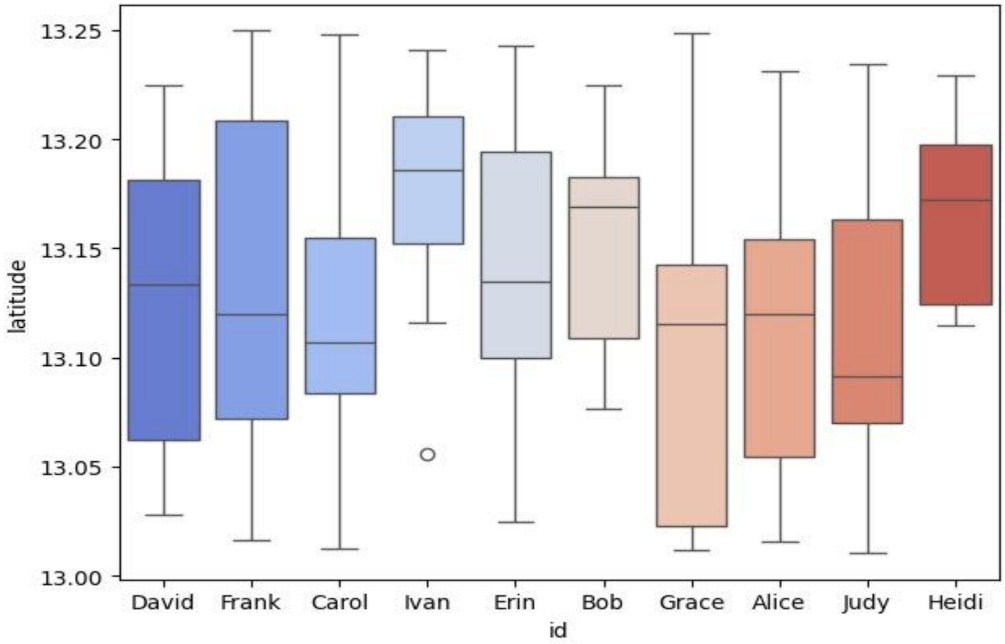


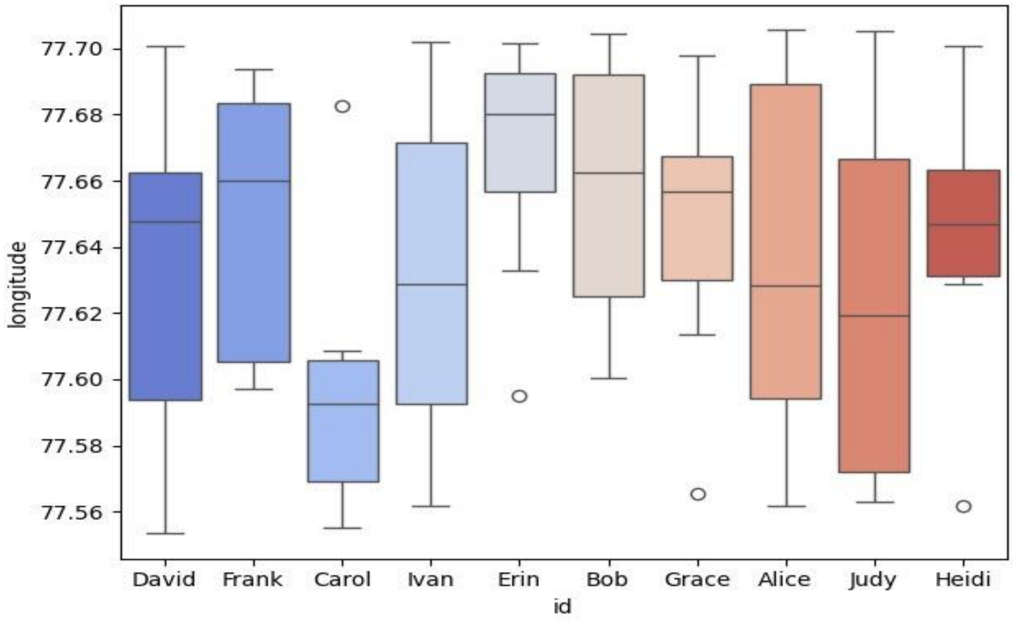
Joint plot :

- This is the joint plot of latitude vs longitude for the dataset.
- This basically describe the continuous plot of the dataset.
- Which means it describes where people have had travelled more,so that it would be easy to predict.



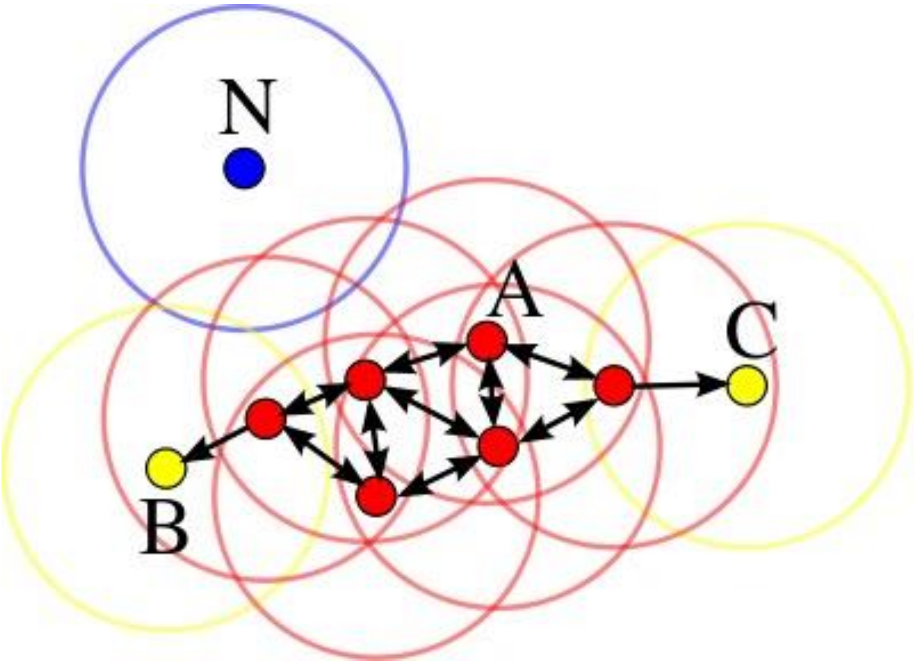
These are the box plots for (latitude,id) and (longitude,id) which basically describes where people have travelled mostly with a mean line associated with in the box.





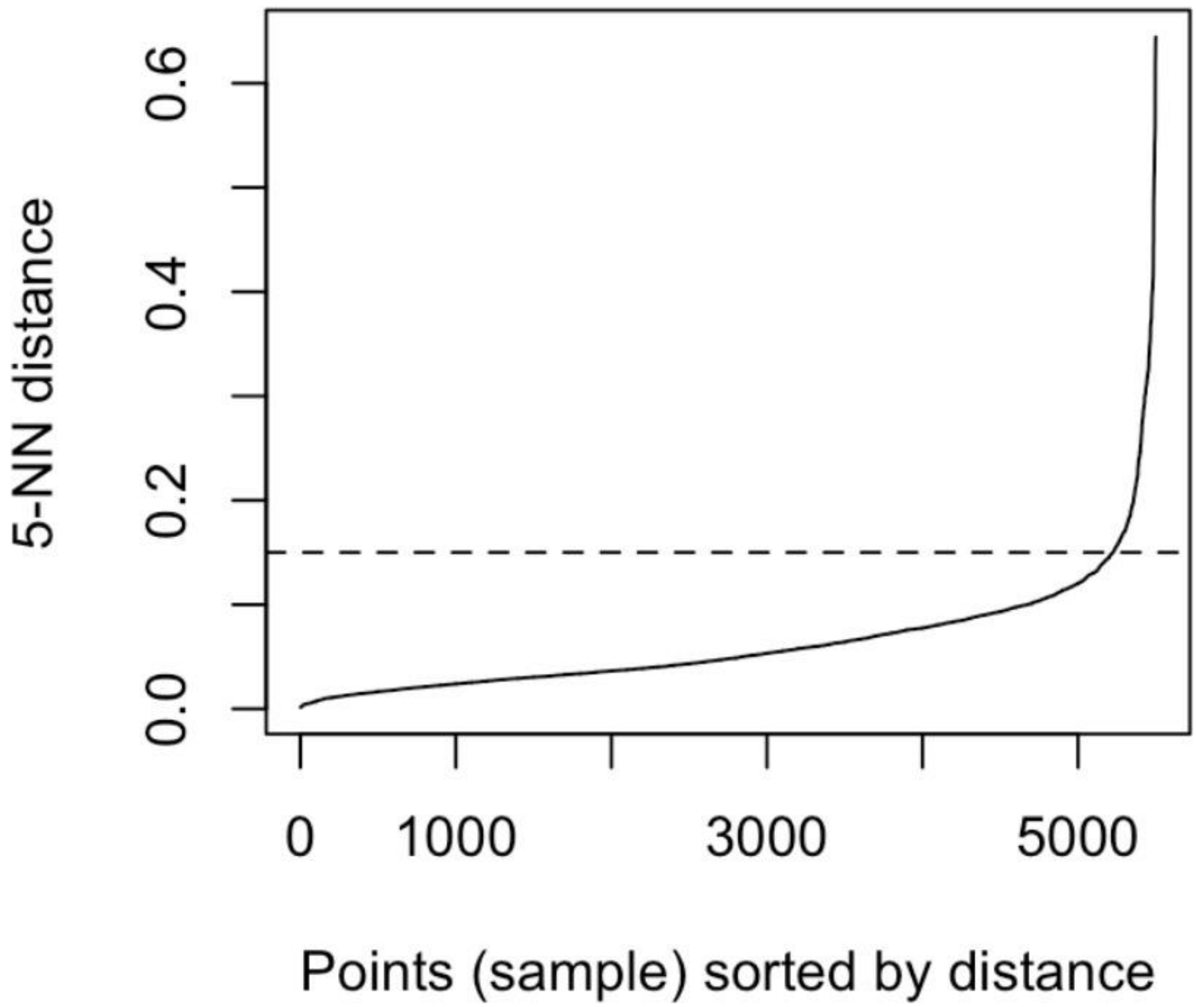
Methodology

- DBSCAN algorithm is utilized to define and establish models parameters.
- This algorithm excels at identifying clusters of varying shapes and densities within a data.
- Once the model is defined, it applied to the data, resulting in the formation of clusters based on the proximity and density of data points.
- Finally the data within each cluster is meticulously filtered to isolate and identify potential infected persons.



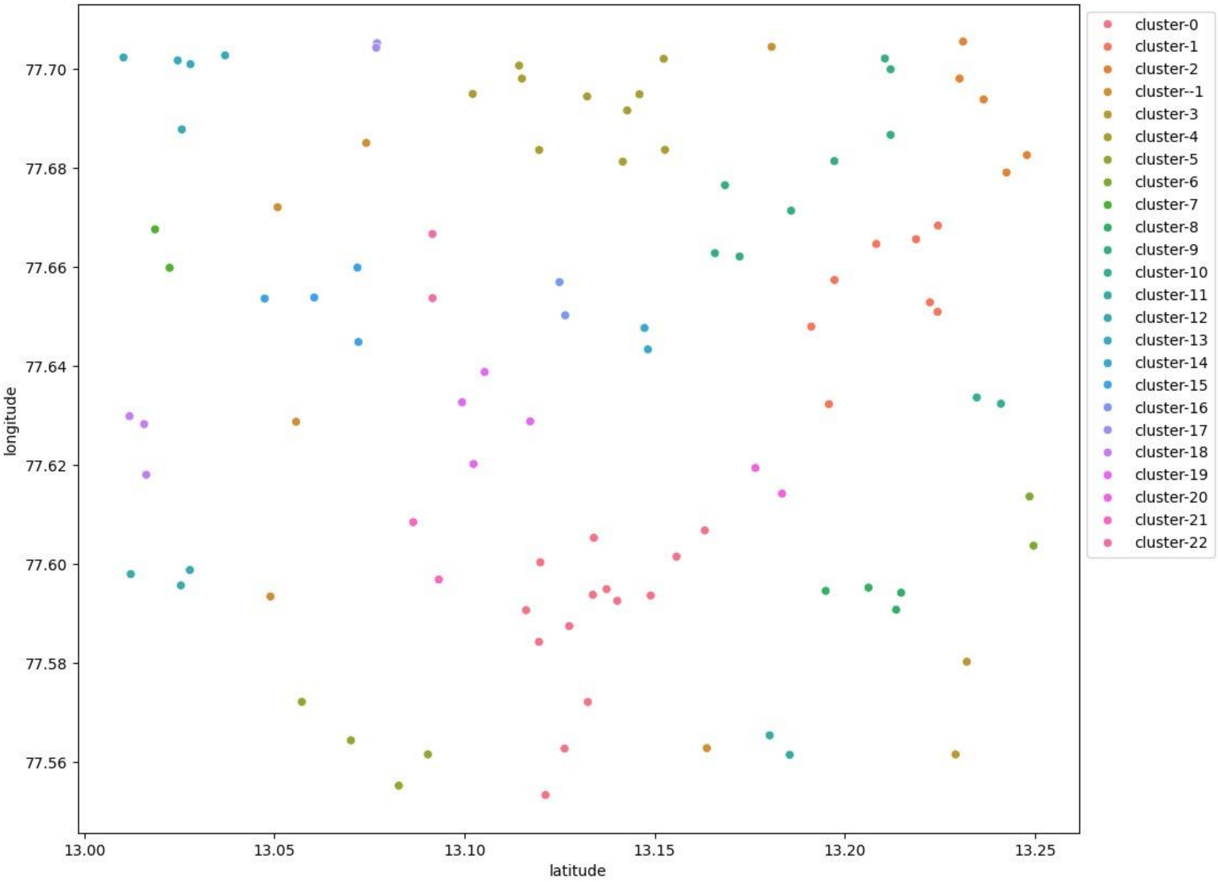
Defining parameters of the model :

- In DBSCAN algorithm the parameter epsilon which is the radius around which it makes cluster.
- Optimal value of the epsilon lies between (0-0.2) as shown.
- It can be seen that the optimal epsilon value is around a distance of 0.15(radians) here.



Reference : https://sefidian.com/2022/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/

- We have decided to check every two people to see if they are within 1.5 kilometers of each other.
- Here we can observe from the plot that people were clustered according to the geographic places they have travelled who might got affected by COVID-19 since geographic space between them is less than 1.5km.
- Here we use Haversine distance for spherical coordinates (appropriate for GPS coordinates)



Working of the algorithm :

1)Creates and fits a DBSCAN model with:
eps=epsilon: Maximum distance between points to be considered neighbors
min_samples=2: Minimum points needed to form a cluster
metric='haversine': Uses haversine distance for spherical coordinates (appropriate for GPS coordinates)
2)Adds cluster labels to the DataFrame. Each point gets assigned a cluster number (-1 means noise/no cluster).
3)Collects all other IDs that are in the same clusters as input_name, excluding noise points (cluster=-1).
4)Finally it returns the persons list and location of the persons to display on the app that is made for this

In [30]:

```
def get_infected_names_and_locations(input_name):
    epsilon = 0.0158288
    model = DBSCAN(eps=epsilon, min_samples=2, metric='haversine').fit(df[['lati
    df['cluster'] = model.labels_.tolist()


    input_name_clusters = []
    infected_data = {}

    for i in range(len(df)):
        if df['id'][i] == input_name:
            if df['cluster'][i] not in input_name_clusters:
                input_name_clusters.append(df['cluster'][i])

    for cluster in input_name_clusters:
        if cluster != -1:
            cluster_data = df[df['cluster'] == cluster]
            for _, row in cluster_data.iterrows():
                if row['id'] != input_name:
                    if row['id'] not in infected_data:
                        infected_data[row['id']] = {
                            'id': row['id'],
                            'latitude': row['latitude'],
                            'longitude': row['longitude']
                        }

    user_location = df[df['id'] == input_name][['latitude', 'longitude']].iloc[0

    infected_list = list(infected_data.values())

    return infected_list, user_location
```

In [32]:

```
get_infected_names_and_locations('David')
```

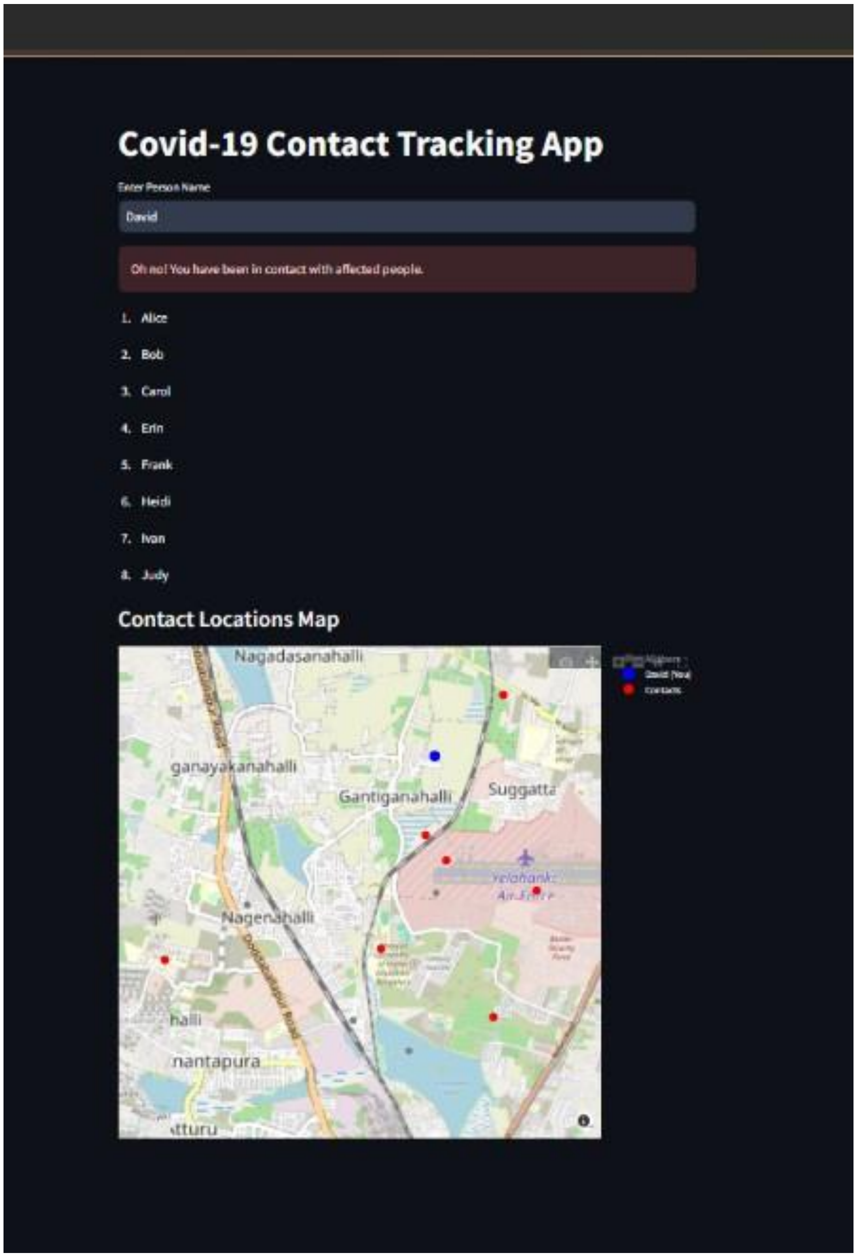
Out[32]:

```
([{'id': 'Judy', 'latitude': 13.126296, 'longitude': 77.5627483},
  {'id': 'Ivan', 'latitude': 13.1401623, 'longitude': 77.5925943},
  {'id': 'Alice', 'latitude': 13.1557417, 'longitude': 77.601504},
  {'id': 'Carol', 'latitude': 13.1275096, 'longitude': 77.5874724},
  {'id': 'Erin', 'latitude': 13.1373459, 'longitude': 77.5949414},
  {'id': 'Bob', 'latitude': 13.1199324, 'longitude': 77.6003416},
  {'id': 'Frank', 'latitude': 13.1339981, 'longitude': 77.6052929},
  {'id': 'Heidi', 'latitude': 13.1972994, 'longitude': 77.6573416}],
 latitude     13.148953
 longitude    77.593651
 Name: 0, dtype: float64)
```

The below function takes the input from the above function and plot it in real time geographical map using plotly library, where other people are represented by "red" and person checking is represented by "blue" as shown below:

Results :

- After deploying it to a web application we can verify by typing a random name.
- If that name exists it applies the told model and predicts whether he/she has been affected or not.
- Else it prompts user to add his/her information to the dataset and then predicts affected or not.



Covid-19 Contact Tracking App

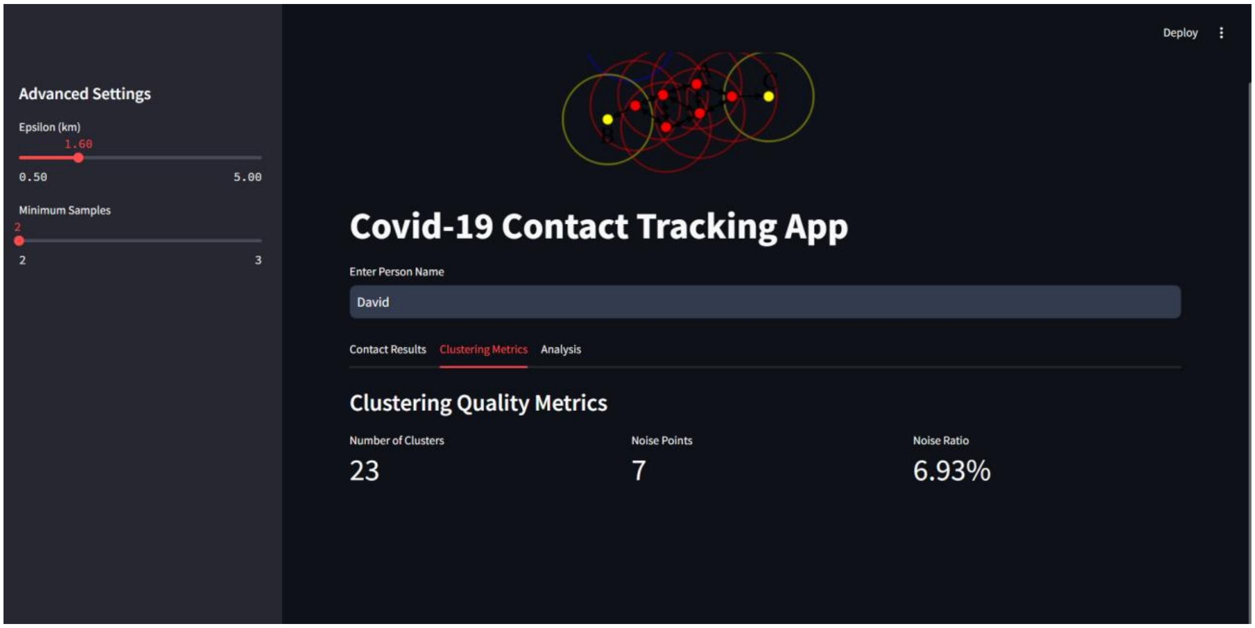
Enter Person Name

David

Oh no! You have been in contact with affected people.

1. Alice
2. Bob
3. Carol
4. Erin
5. Frank
6. Heidi
7. Ivan
8. Judy

Contact Locations Map



Advanced Settings

Epsilon (km)

1.60

0.50 5.00

Minimum Samples

2

2 3

Covid-19 Contact Tracking App

Enter Person Name

David

Clustering Quality Metrics

Number of Clusters	Noise Points	Noise Ratio
23	7	6.93%