ARTIFICIAL INTELLIGENCE

COURSE ASSIGNMENT - 1

COVID -19 Patient Contact Tracing using Machine Learning

Submitted By:

KAWALJEET SINGH BATRA
Roll: **2019-IMG-027**

Submitted To:

Dr. PINKU RANJAN

INTRODUCTION

Contact tracing is a process that is widely used by public and private health ministries and organisations to help stop spread of some infectious disease, eg COVID-19, within a community. Here, I propose a Machine Learning based approach to achieve contact tracing of infected COVID-19 patients and detection of contamination spread region.

HOW CONTACT TRACING WORKS?

Once a person is positive for coronavirus (SARS - Covid-19), it is very important to identify other people who may have been infected by the patients diagnosed. To identify infected people, the authorities follow the activity of patients diagnosed in the last 14 days. This process is called contact tracing. Depending on the country and the local authority, the search for contacts is carried out either by manual methods or by numerical methods.

In this project, I will be proposing a digital contact tracing algorithm that relies on GPS data, which can be used in contact tracing with machine learning

CONTACT TRACING WITH MACHINE LEARNING

DBSCAN is a density-based data clustering algorithm that groups data points in a given space. The DBSCAN algorithm groups data points close to each other and marks outlier data points as noise. I will use the DBSCAN algorithm for the task of contact tracing with Machine Learning.

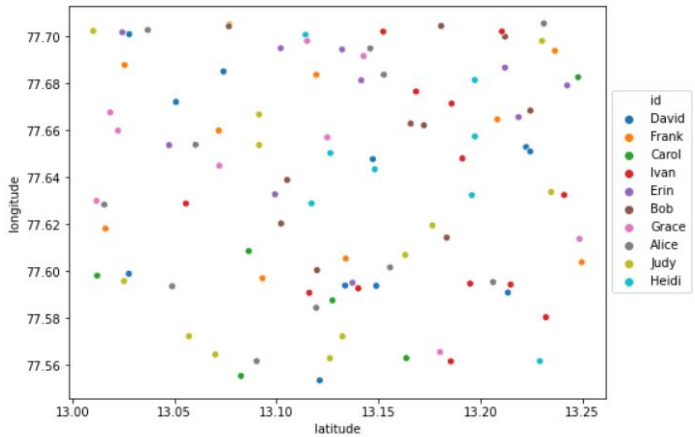
The Dataset that we will be using for this demonstration, is a free open-source JSON data available. It can be downloaded from here.

We need to download this data and save it as .json file in our project folder.

PREPARING THE DATASET AND ANALYSING THE DATA POINTS

We will be using Numpy, Pandas and Seaborn to preprocess our data and then visualize it using scatter plot and box plot to draw inferences.

```
: ## plotting a scatter plot to visualize the data points
plt.figure(figsize=(8,6))
sns.scatterplot(x="latitude", y="longitude", data=df, hue="id")
plt.legend(bbox_to_anchor= [1, 0.8])
plt.show()
```



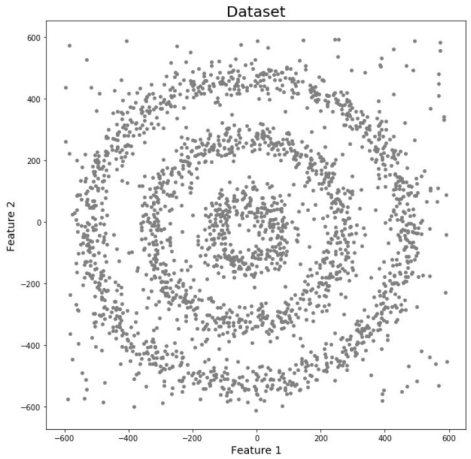
HOW DBSCAN CLUSTERING WORKS?

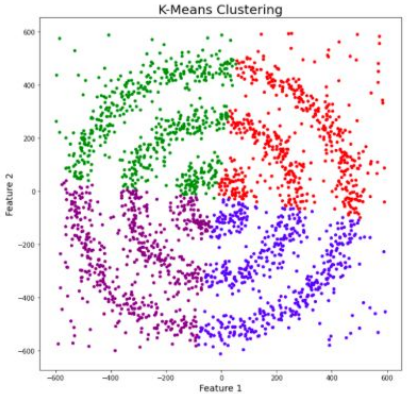
DBSCAN Algorithm groups '**densely grouped**' data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. The most exciting feature of DBSCAN clustering is that it is robust to outliers. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

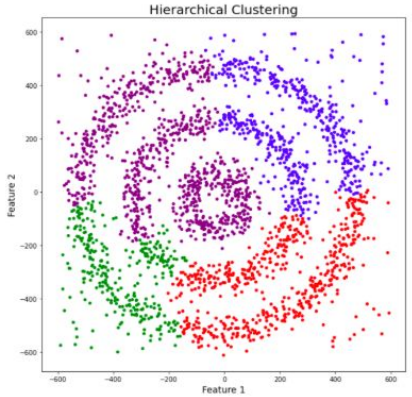
DBSCAN requires only two parameters: **epsilon** and **minPoints**. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

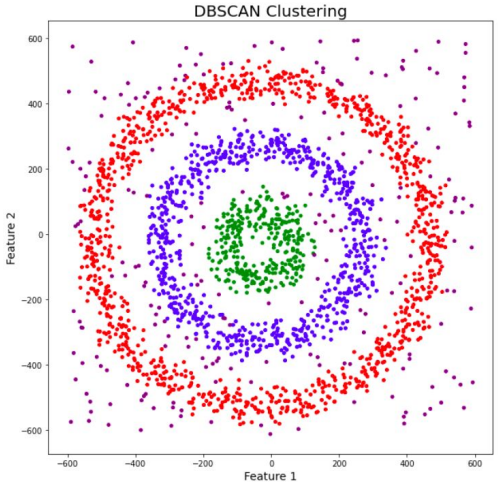
DIFFERENT FROM K-MEANS ?

K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering.









PREPARING THE DBSCAN CLUSTERING MODEL

Now, preparing the Clustering model to cluster infected people based on their latitudes and longitudes.

Preparing the DBScan Model

```
def find_infected_persons(name):
    epsilon = 0.0018288
    model = DBSCAN(eps=epsilon, min_samples=2, metric='haversine').fit(df[['latitude', 'longitude']])
    df['cluster'] = model.labels_.tolist()

    input_name_clusters = []
    for i in range(len(df)):
        if df['id'][i] == name:
            if df['cluster'][i] in input_name_clusters:
                pass
            else:
                input_name_clusters.append(df['cluster'][i])

    infected_names = []

    for cluster in input_name_clusters:
        if cluster != -1:
            ids_in_cluster = df.loc[df['cluster'] == cluster, 'id']
            for i in range(len(ids_in_cluster)):
                member_id = ids_in_cluster.iloc[i]
                if (member_id not in infected_names) and (member_id != name):
                    infected_names.append(member_id)
                else:
                    pass



    return infected_names
```

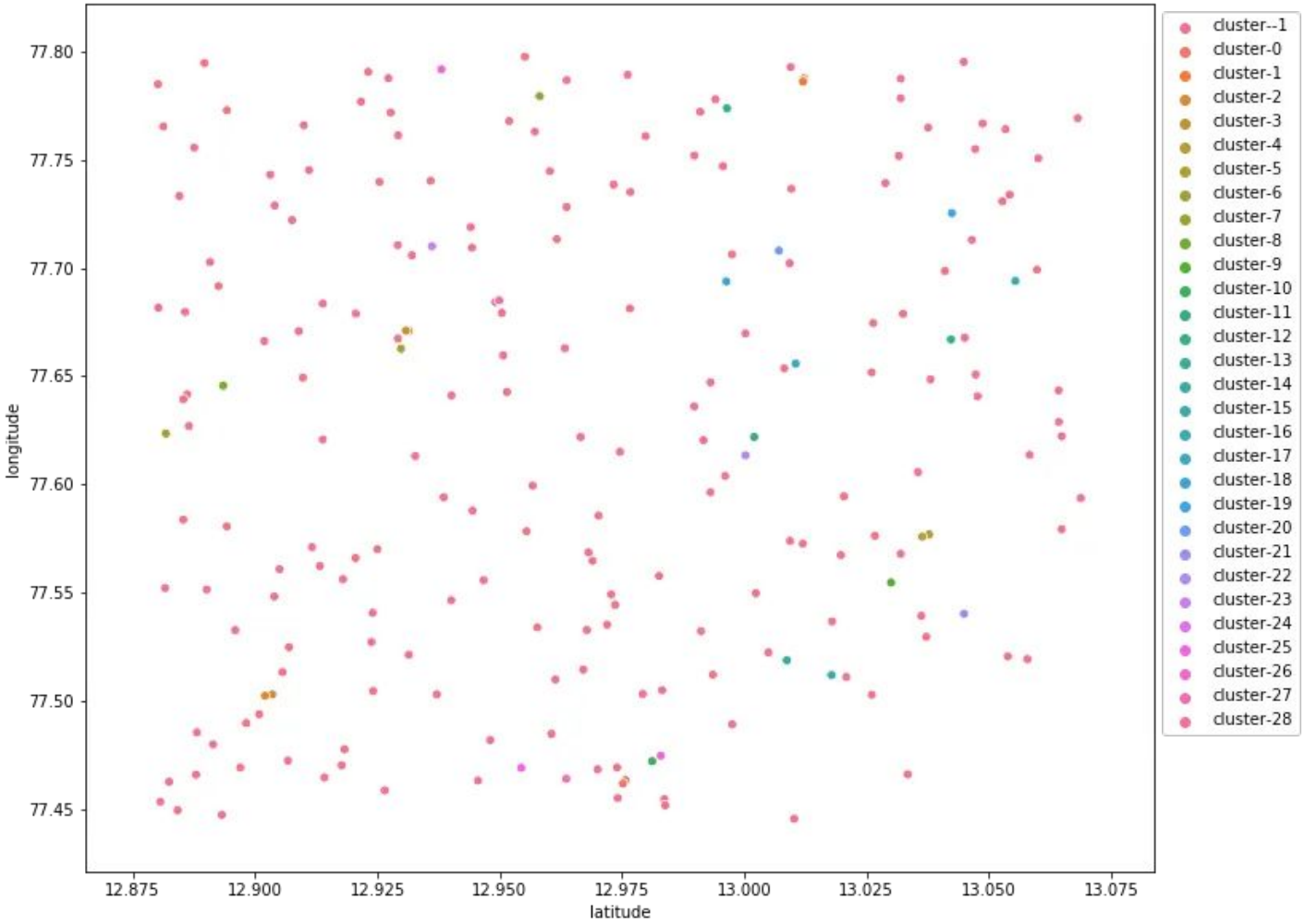
PREDICTING THE INFECTED PERSONS

To find people who may be infected by the patient, we'll just call the **find_infected_persons** function and enter a name from the dataset as a parameter.

The function will return the name of the person who is probabilistically infected by the person whose name is passed as an argument.

Eg:

```
In [23]: print(find_infected_persons("Bob"))
         ['Judy']
```



Clusters depicting infected persons spread over the latitude and longitude grid.

CONCLUSION

From the above computed results, we can say that a clustering algorithm like DBSCAN can perform data point clustering without prior knowledge of the datasets.

This technique, is hence useful for detection of the people who can potentially act as infection carriers and also be useful for prediction of potential Covid-19 hotspots and contamination zones.

IMPORTANT LINKS

The Code files for this Assignment can be found here.

Packages used:

- Python 3.8
- Scikit Learn - DBSCAN Clustering model
- Numpy
- Pandas
- Matplotlib

Jupyter Notebook was used for compiling the code for this Assignment.

THANK YOU