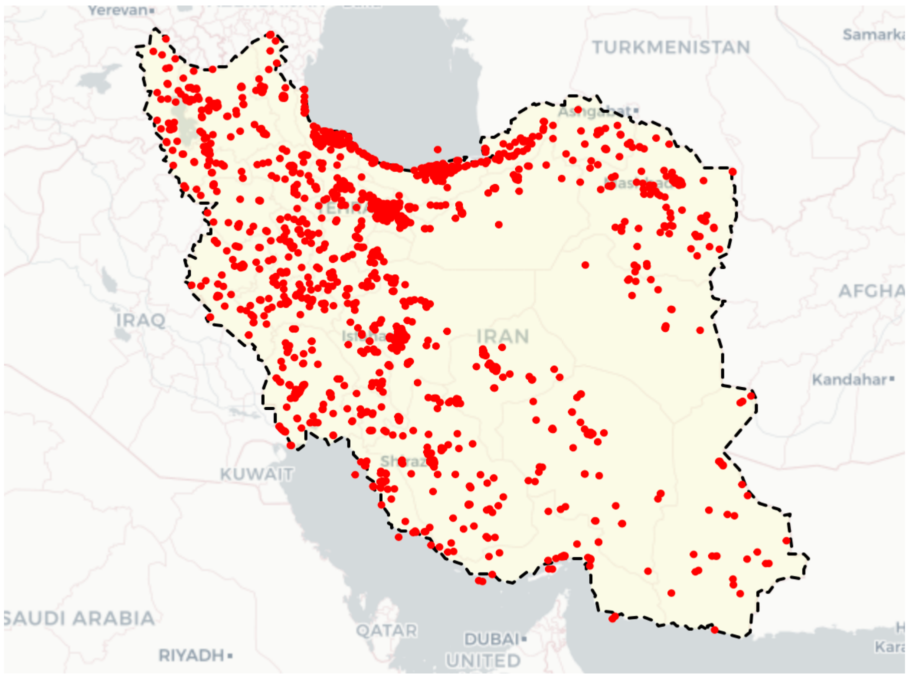
Implementation 2: DBSCAN

In the previous part, you got introduced to several inherent weaknesses of the K-means algorithm.
In this part, you start working with more sophisticated algorithm, called DBSCAN, which can somewhat handle problems mentioned above. Since DBSCAN is a little complicated you don’t need to implement the algorithm in this part, and you can use available libraries (DBSCAN library in sklearn package is recommended) to complete this part.
Also, it is strongly recommended to use Jupyter notebooks to complete this part.



in this assignment, you've been supplied with geographical distribution of COVID-19 patients inside Iran (covid.csv). each row in the csv file represents a detected COVID-19 case, first column is the geographical latitude and second column is the geographical longitude.
for better visualization of geographical data we're using a python library called "folium".

You can install folium using pip: `pip install folium` here is an example code for displaying a geolocation using folium:

```
import folium

# set iran as map starting point
m = folium.Map(location=[32.427910, 53.688046], zoom_start=5)

# mark an example location
loc = [35.703136,51.409126]
folium.Marker(location=loc).add_to(m)
```

for more information on folium see here.

As you already know DBSCAN requires 2 main parameters, eps and minPts.
eps is the maximum allowed distance between 2 data points in the same cluster, and minPts is the minimum number of data points to create a cluster. Using these two important parameters DBSCAN dynamically creates as many as needed clusters while discarding any clusters without enough data points and keeping the outliner data points out of any valid cluster. As a result, changing the value of these to parameters changes the final number of clusters and the way the algorithm works.
Our purpose is to use DBSCAN to find dense disease clusters inside Iran.

A) Load the dataset from csv file and plot it on map using folium. (for better visualization use folium circles)

```
folium.Circle(location=loc,
              radius=1,
              color="red",
              fill=True).add_to(m)
```

B) Run DBSCAN algorithm with arbitrary values for eps and minPts.

C) Fine-tune eps and minPts parameters so that each cluster only includes patients from heavily infected areas. these clusters must exclude outlier locations (Hint: Tehran and Qom are possible dense clusters)

D) Plot each cluster on map using a different color. Plot outliers with same color.