Clustering in Geo Social Networks

Done By:

Balarammahanthi Gautham(181CO212)

V V Sai Tarun(181CO257)

Clustering In Geo Social Networks

Clustering is a common task of data mining, which divides a set of objects into groups such that objects in the same group (called a cluster) are similar to each other while objects in different clusters are dissimilar. Clustering finds applications in machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics. Specific applications include grouping homologous sequences into gene families in bioinformatics, partitioning the general population of consumers into groups in market research, recognizing communities within large groups of people in social networks, dividing a digital image into distinct regions for border detection or object recognition. Clustering can be achieved by various algorithms that may differ significantly in how they define clusters.Popular definitions of clusters are groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. The distance function, the density threshold or the number of expected clusters to use depend on the data to be clustered and the intended use of the clustering results.

CLUSTERING:

Clusters are dense regions in the data space, separated by regions of the lower density of points.Clustering is a method in Unsupervised machine learning.Clustering is a common task of data mining, which divides a set of objects into groups.Objects in same group (called a cluster) are similar to each other while objects in different clusters are dissimilar.Clustering has various applications in machine learning, pattern recognition, image analysis.

GEO SOCIAL NETWORKS :

Geosocial networking is a type of social networking in which geographic services and capabilities such as geocoding and geotagging are used to enable additional social dynamics.User-submitted location data or geolocation techniques can allow social networks to connect and coordinate users with local people or events that match their interests.In

geo social networks people can share their geographic location with other users.This sharing process is called “Check in”. Check in is a triplet containing (user, place, time).

Clustering in Geo social networks is done based on the check in data for various places.It is done using the DBSCAN(Density Based Spatial Clustering of Application with Noise) algorithm.Spatial Clustering is done first based on the longitudes and latitudes of different places.Later we are going to calculate social distance between every two place in each clusters obtained.Finally we are going to apply some constraints on social distance to obtain the final Geo Social Clusters.

DBSCAN Algorithm :

Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

DBSCAN algorithm requires two parameters :

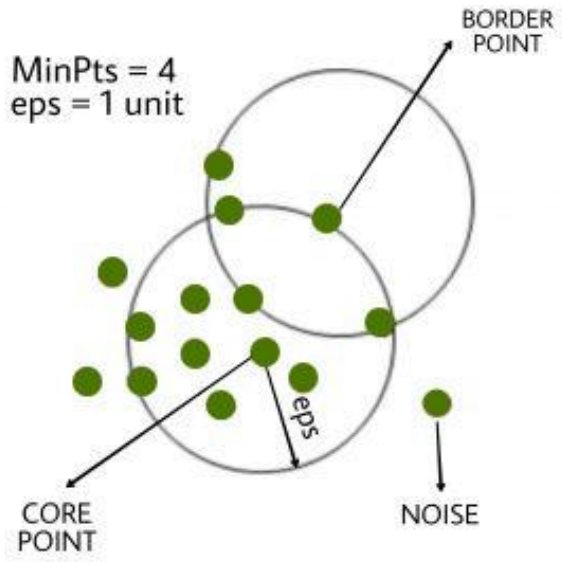
1. **epsilon** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered as neighbors.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen.

In this algorithm, we have 3 types of data points.

1.Core Point: A point is a core point if it has more than MinPts points within eps.

2.Border Point: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.

3.Noise or outlier: A point which is not a core point or border point.



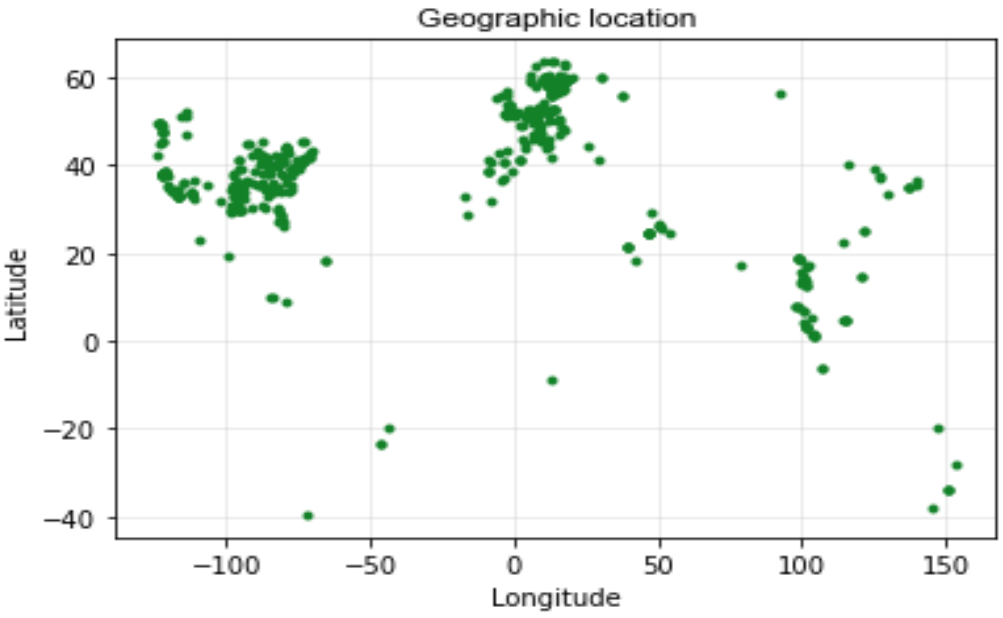
DBSCAN Algorithm:

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.

   A point *a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points *a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is neighbor of *c*, *c* is neighbor of *d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of *a*.
4. Iterate through the remaining unvisited points in the dataset.

Gathering the Data:

Friendship data and location data.Check-in data will be having the attributes user id,placeid,date and time which gives the information when a particular user visits a particular place.User Friendship data have only two attributes user_id 1,userid2 which helps to calculate the social distance between places.Location data gives the longitude and latitudes of all the place id which will be useful of initial place clustering.We had cleaned the datasets and merge check-in data and Location data and had removed the duplicates to obtain the final required dataset.We can see the plot of the collected data below.



Place clustering Using DBSCAN:

After getting the final sample dataset we are going to find the clusters based on their spatial distance using their latitudes and longitudes.For clustering we have used the DBSCAN model which is already there in the sklearn module with an epsilon value 0.25 and minimum points of 5 which is mentioned in the reference paper.After applying the clustering model we get a clusters_df data frame which contains all the clusters and outliers_df data frame which contains all the outliers.Now we are plotting the graph using the two dataframes we have obtained and model.labels which will obtained from the DBSCAN model.This is the output we had obtained after applying the DBSCAN model for the final dataset.First line give number of places are there in each cluster and counter with -1 gives the outliers count and the dataframe below gives the outiers_df.

```
Counter({1: 198, 0: 185, 2: 33, -1: 24, 3: 21, 4: 15, 6: 7, 7: 6, 8: 6, 5: 5})
          placeid         lng        lat
34649264  7078193   37.570946  55.760597
25910517  1318327   18.398269  69.772862
16385560   492242  114.152087  22.286604
3170071     37200  -99.095135  19.405210
35449177  7395262  174.776715 -41.284212
Number of clusters =9
```

The above picture shows the number of clusters that are present in the above graph and the data frame gives the outliers.

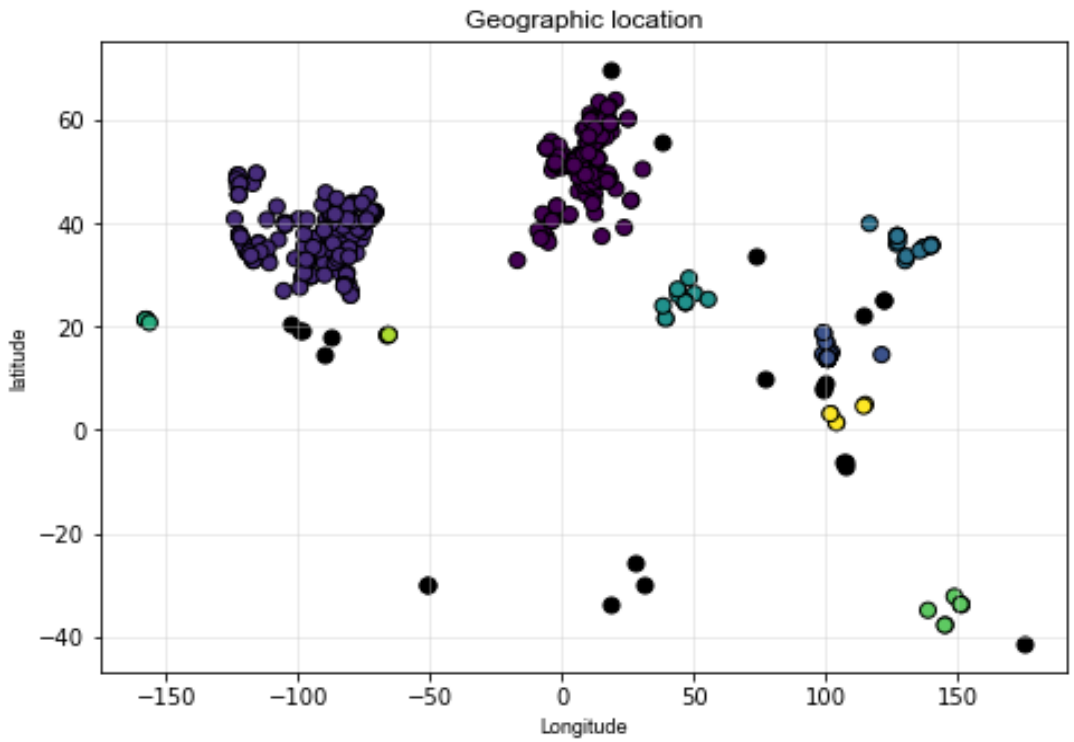
Python code for Place clustering :

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
from collections import Counter
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline

data=pd.read_csv('gowalla_checkins.csv')
del data['datetime']
d=pd.read_csv('gowalla_friendship.csv')
e=pd.read_csv('gowalla_spots_subset1.csv')
e=e.rename(columns={"id":"placeid"})
e.drop(e.columns[[1, 4,5,7,8,9,10,11]], axis = 1, inplace = True)
balanced_data=pd.merge(e,data, on = "placeid", how = "inner")
balanced_data=pd.DataFrame.drop_duplicates(balanced_data)
datal=balanced_data.drop(balanced_data.columns[[3,4]], axis = 1)
place_clustering_data=pd.DataFrame.drop_duplicates(datal)
place_clustering_data.head()
data=place_clustering_data.sample(500)

print('Geographic location plot')

_=plt.plot(data['lng'],data['lat'],marker='.',linewidth=0,color='#128128')
_=plt.grid(which='major',color='#cccccc',alpha=0.45)
_=plt.title('Geographic location',family='Arial',fontsize=12)
_=plt.xlabel('Longitude')
_=plt.ylabel('Latitude')

dbscan_data=data[['lng','lat']]
dbscan_data=dbscan_data.values.astype('float32',copy=False)
dbscan_data_scaler=StandardScaler().fit(dbscan_data)
dbscan_data=dbscan_data_scaler.transform(dbscan_data)
model=DBSCAN(eps=0.25,min_samples=5,metric='euclidean').fit(dbscan_data)
model
outliers_df=data[model.labels_==-1]
clusters_df=data[model.labels_!=-1]
colors=model.labels_
colors_clusters=colors[colors!=-1]
color_outliers='black'
clusters=Counter(model.labels_)
print(clusters)
print(data[model.labels_==-1].head())
print('Number of clusters ={}'.format(len(clusters)-1))
```

```
fig=plt.figure()
ax=fig.add_axes([.1,.1,1,1])
ax.scatter(clusters_df['lng'],clusters_df['lat'],c=colors_clusters,edgecolors='black',s=50)
ax.scatter(outliers_df['lng'],outliers_df['lat'],c=color_outliers,edgecolors='black',s=50)
ax.set_xlabel('Longitude',family='Arial',fontsize=9)
ax.set_ylabel('latitude',family='Arial',fontsize=9)
plt.title('Geographic location',family='Arial',fontsize=12)
plt.grid(which='major',color='#cccccc',alpha=0.45)
plt.show()
```

Output:

Output is the place clustering plot for the selected 500 places in a certain location.We can see each cluster is colored with different colors and outliers are colored with black color.



Social Distance calculation:

After getting the place clusters we need to find the social distance between every two places in each cluster that we obtained(We can exclude all the outliers).Social distance is calculated by the given equation:

$$DS=|CU|/(u1 \cup u2)$$

where CU is number of contributing users for the places and u1 and U2 are the users visited places P1 and P2.To calculate U1 and U2 we need to develop a set from the check-in dataset by appending all the unique users visited places P1 and P2 irrespective of date and time.

After calculating the sets U1 and U2 we need to calculate the contributing users.Contributing users means if a user visits both places or if user1 visited place “1” and user2 visited place “2” and if user1 and user2 are friends then they are also contributing users.CU can be calculated using the User Friendship dataframe and the sets U1,U2.Union of U1 and U2 is equal to U1+U2-Intersection of(sets U1,U2).After calculating all the parameters between two places we calculate the social distance between these two places.The value of social distance between any two places lies 0 and 1.

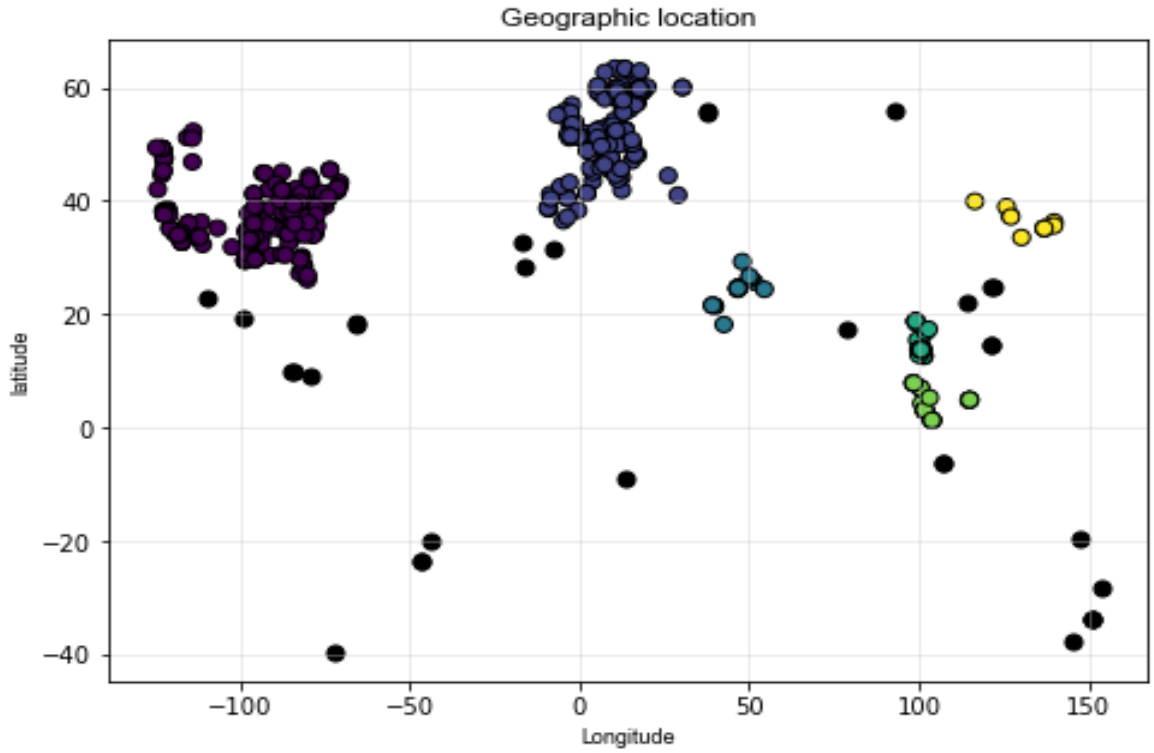
Python code for Social distance calculation:

```
def calculate_contributing_users(p1,p2):
    cu=0
    for i in p1:
        l=d[d['userid1']==i]
        r=l['userid2'].values
        q=set(r).intersection(p2)
        cu=cu+len(q)
    return cu

def check_social_distance(df,new_clusters_df):
    p1=[]
    p2=[]
    for i in df['placeid']:
```

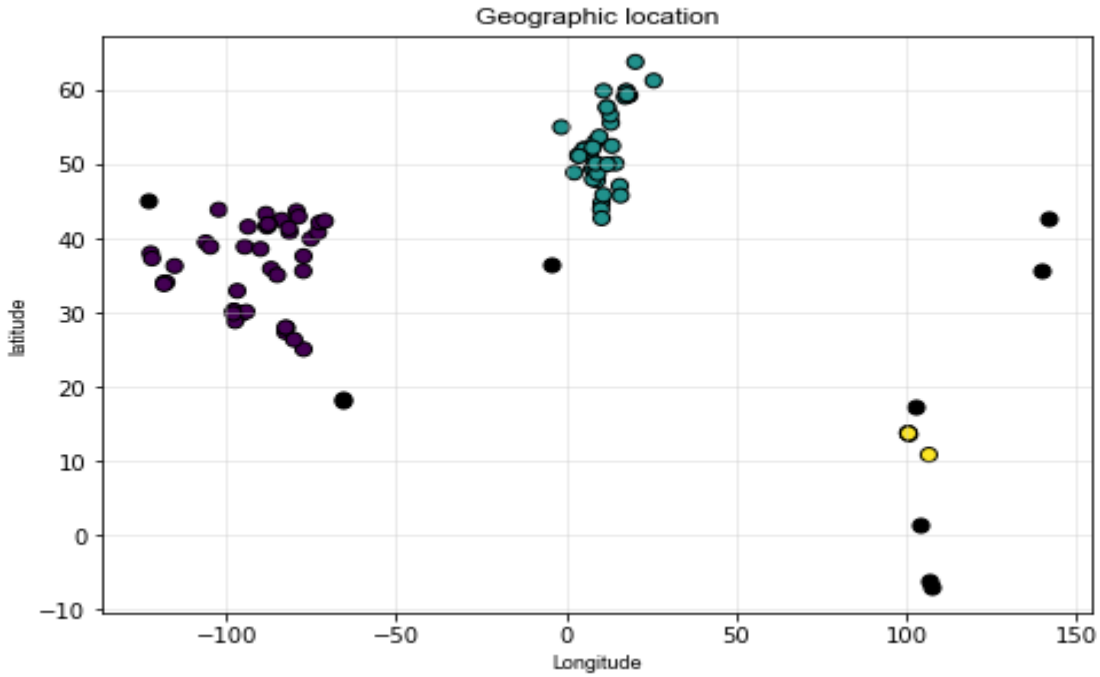
```
    f=balanced_data[balanced_data['placeid']==i]
    p1=f['userid'].values
    sd=[]
    mean=0
    for j in df['placeid']:
        if(i!=j):
            r=balanced_data[balanced_data['placeid']==j]
            p2=r['userid'].values
            k=set(p1).intersection(p2)
            union_p1_p2=len(p1)+len(p2)-len(k)
            q1=set(p1)
            q2=set(p2)
            q1=q1-k
            q2=q2-k
            contributing_users=calculate_contributing_users(q1,q2)
            social_distance=(contributing_users+len(k))/union_p1_p2
            sd.append(social_distance)
            mean=mean+social_distance
    mean_sd=mean/len(df['placeid])
    if(mean_sd<0.05):
        df=df.drop(df[df.placeid==i].index)
return len(df),df
```

Active Social Clusters :

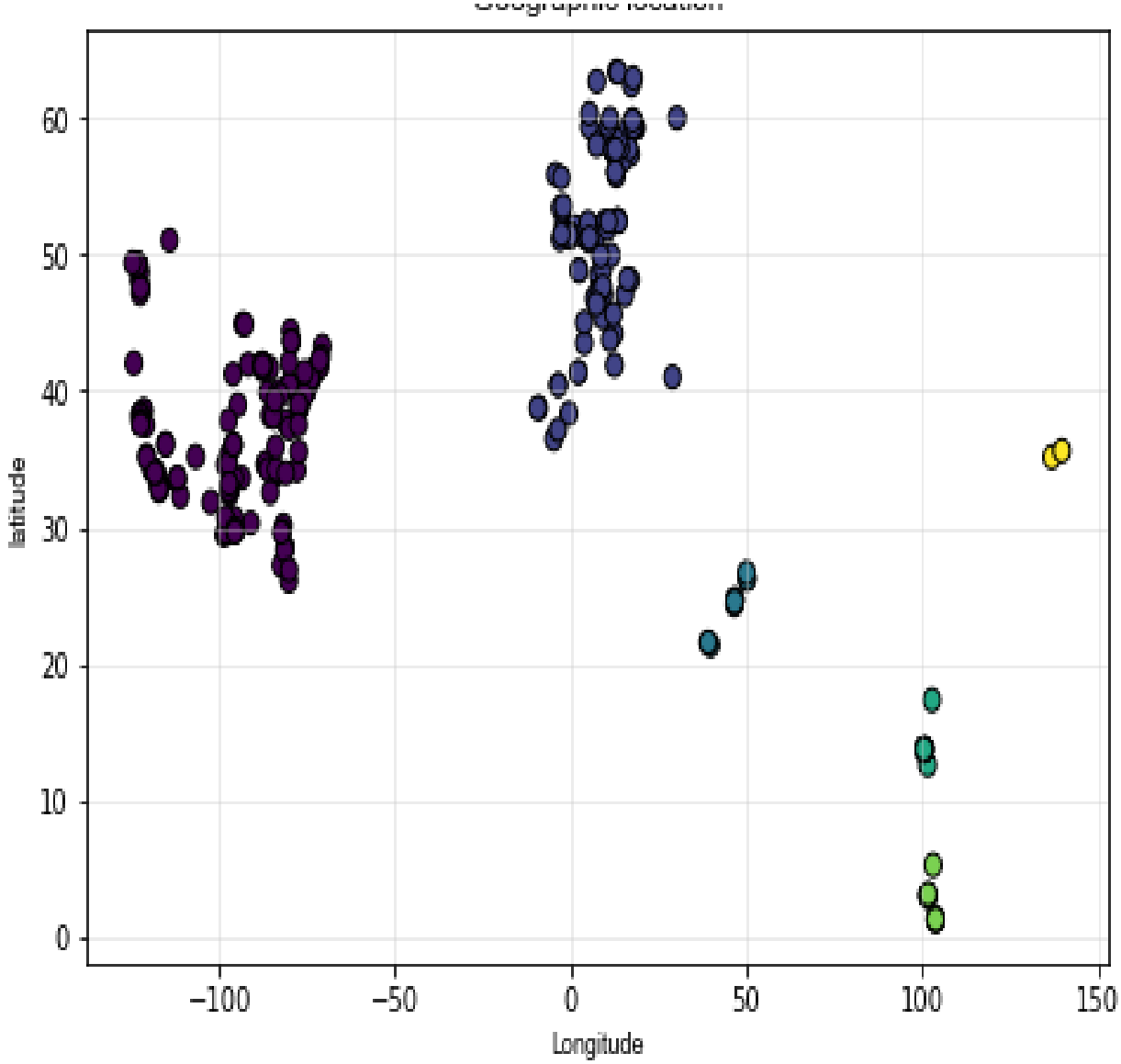
In order to find Active Social clusters we need to find mean of social distance of every point in each cluster with respective all the other points in the same cluster.We are going to define this value as ℨ.We are going to apply various value constraints to ℨ to obtain different plots which will be discussed later.We remove the places whose mean social distance value is less than ℨ from the place clusters that we obtained before.After obtaining new cluster points we are going to plot them in a new plot with a different ℨ values.



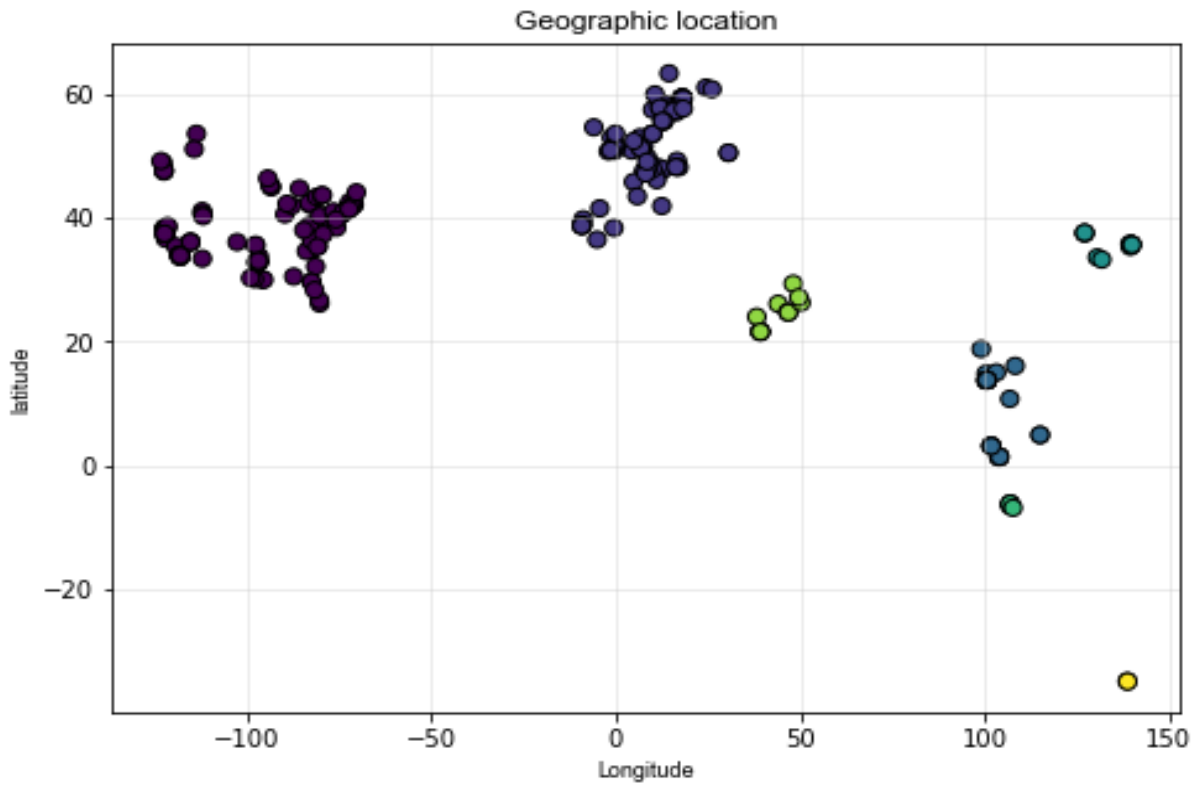
Place clustered data for 500 places



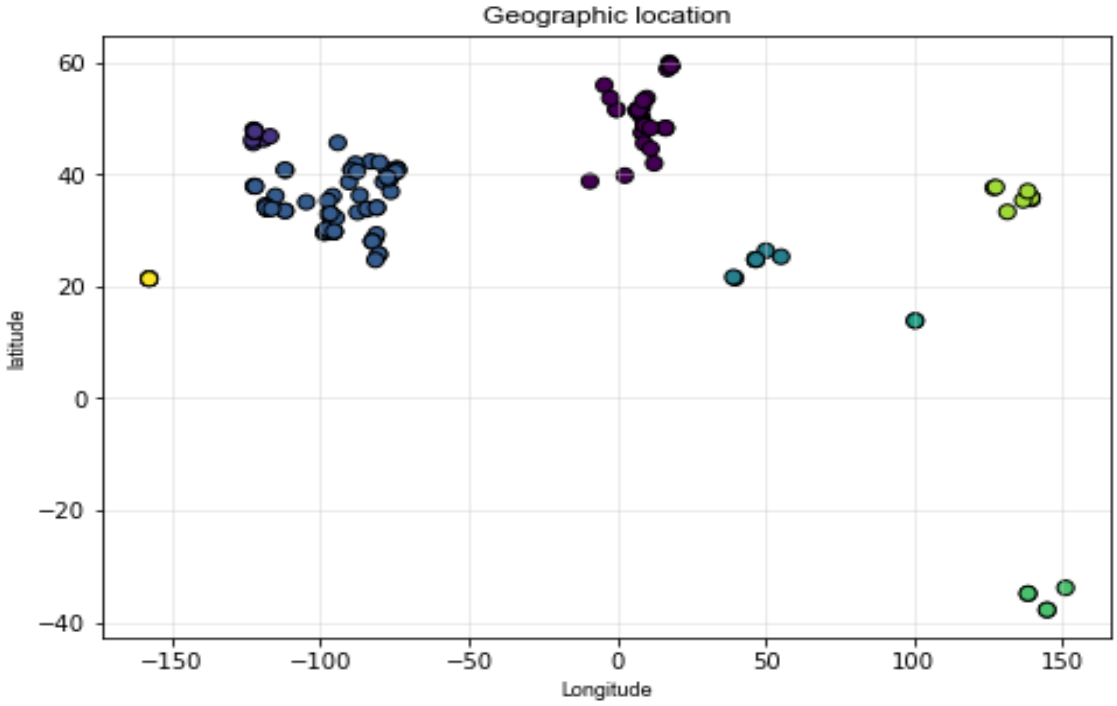
Place Clustering for 100 places



ƨ greater than 0.001

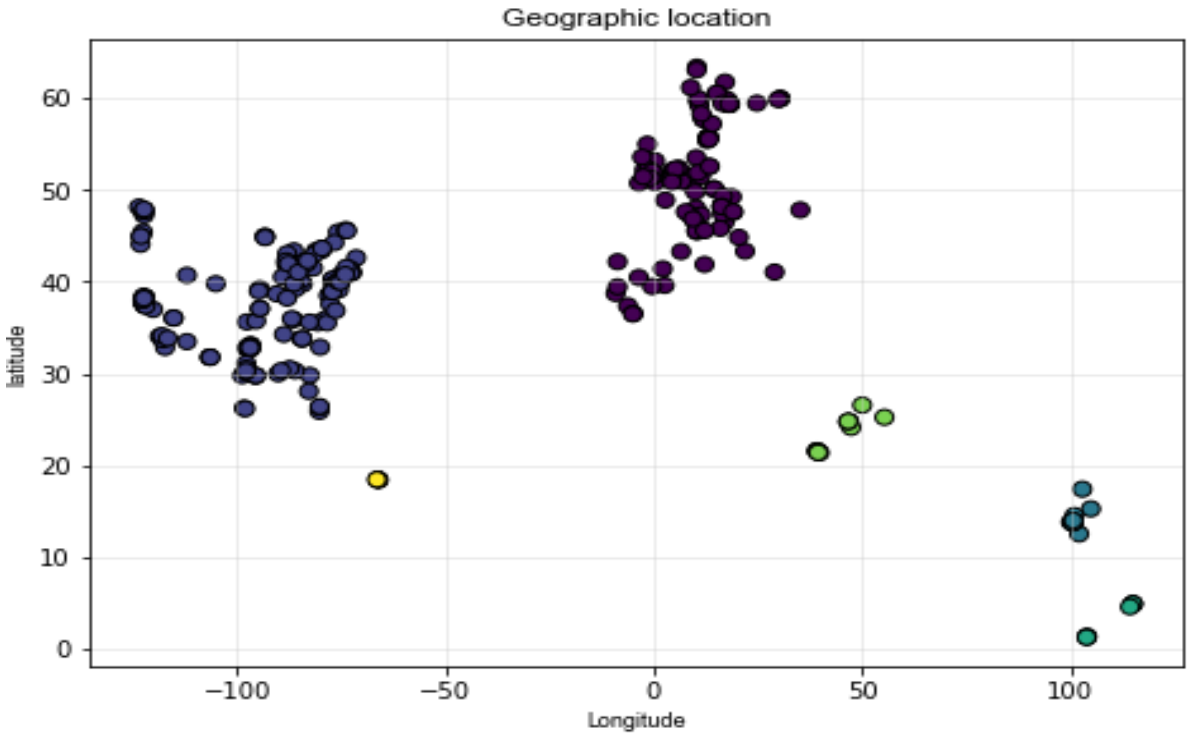


ϖ greater than 0.002



ϖ greater than 0.01

We can see as the value of ℸ increases the length of each clusters decreases as shown in the above plots if the value of ℸ decreases length of clusters increases as shown below.



ℸ greater than 0.0001

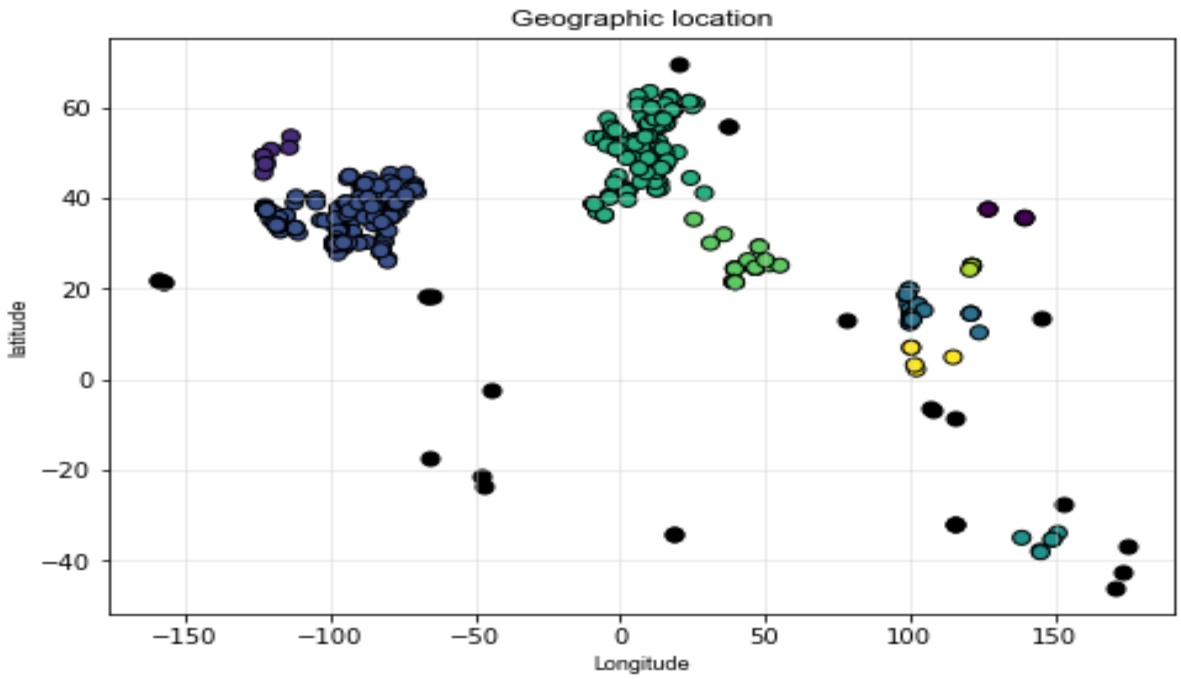
So we can conclude that size of clusters is directly proportional to the value of ℸ .We can increase the value of ℸ to get more socially connected places and we can decrease the size of ℸ to get more dense clusters.Now we are going to see how to get strong communities in which every place in a particular cluster has a social distance greater than zero.

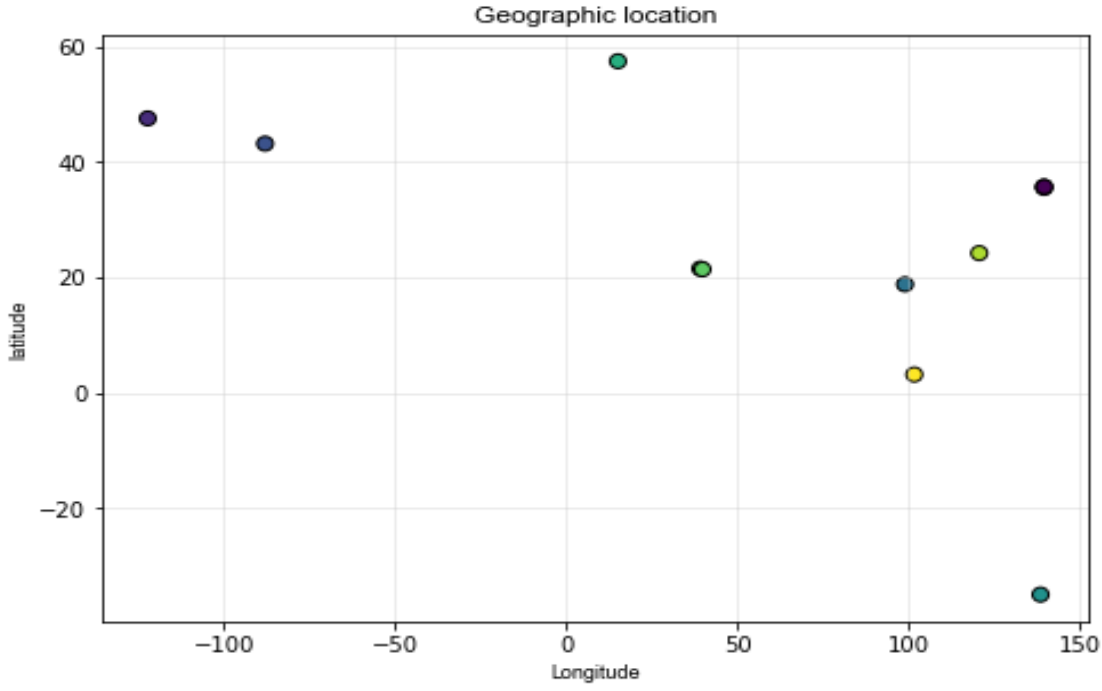
STRONG COMMUNITIES:

In order to find the strong communities we are going to calculate the social distance between every two points in each cluster that we obtained by doing place clustering.And we will include all the places whose social distance is not equal to zero with any place in the same cluster.The number of places in the strong communities is much lower than compared to place clustering plot because it is a hard constraint which will eliminate most of the points.

Python code to find Strong Communities:

```
def check_social_distance(df,new_clusters_df):
    p1=[]
    p2=[]
    for i in df['placeid']:
        f=balanced_data[balanced_data['placeid']==i]
        p1=f['userid'].values
        sd=[]
        mean=0
        for j in df['placeid']:
            if(i!=j):
                r=balanced_data[balanced_data['placeid']==j]
                p2=r['userid'].values
                k=set(p1).intersection(p2)
                union_p1_p2=len(p1)+len(p2)-len(k)
                q1=set(p1)
                q2=set(p2)
                q1=q1-k
                q2=q2-k
                contributing_users=calculate_contributing_users(q1,q2)
                social_distance=(contributing_users+len(k))/union_p1_p2
                sd.append(social_distance)
                mean=mean+social_distance
        mean_sd=mean/len(df['placeid'])
        for j in sd:
            if(j==0):
                df=df.drop(df[df.placeid==i].index)
                break
        del sd
    return len(df),df

def calculate_contributing_users(p1,p2):
    cu=0
    for i in p1:
        l=d[d['userid1']==i]
        r=l['userid2'].values
        q=set(r).intersection(p2)
        cu=cu+len(q)

    return cu
```





Strong Community places in each cluster

We can see very few points in the above plot because we have considered only 500 places only if we consider more places in the location

then we can see more points in each cluster.We can determine most socially connected places in a certain location and it helps a lot for many purposes. These places are the ones which are connected to almost all the points in a particular place with social distance greater than zero with all the places in a particular location.

```
In [43]: new_clusters_df
```

Out[43]:

	placeid	lng	lat
31726101	6443997	139.560653	35.703955
19365767	699563	139.632872	35.500963
33218666	6727583	139.695645	35.662122
16856472	522003	-122.360589	47.524538
12062532	269054	-88.053632	43.153736
33576087	6802894	98.967979	18.797715
8461583	136144	138.598087	-34.923836
31906118	6473993	15.082230	57.419477
27950227	1594696	39.151640	21.523811
29449171	3767748	39.791211	21.396737
30877882	5975969	120.684922	24.198337
22062030	919089	101.691440	3.175318

```
In [44]: labels
```

```
Out[44]: [0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 7, 8]
```

Strong Community points

Contribution:

We get to know what is geo social clustering and terminologies related to the topic like social distance,spatial distance etc.. from the reference paper.Unlike given in the paper we have developed a completely different model.We took the idea of calculation of all the needed elements from the paper unlike they have used to calculate the geo social distance for every point where as we have calculated only social distance and mean social distance for every point in a particular location.To get all the places in a particular location we have applied DBSCAN place clustering using the check in data and then we have calculated the mean social distance of each place with all the places in a particular cluster.We had plotted different plots for different values of ℨ.We had plotted the strong community members in every cluster which is not mentioned in the paper.

References:

1. http://www.cs.uoi.gr/~nikos/TKDEGeosocialClustering.pdf
2. https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/
3. https://developers.google.com/machine-learning/clustering/overview
4. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
5. https://sklearn.org/modules/generated/sklearn.cluster.dbscan.html

Done By:

Balarammahanthi Gautham(181CO212)

V V Sai Tarun(181CO257)