Location Data – Battle of Neighborhoods

Prashant Hegde

April 26, 2020

1. Introduction

Introduction: Business Problem

Just like we are taught about how to use location data, I can think of a scenario where a person is moving from one place to another for a job change or some other reason. I have been living in Bangalore, but after completing this data science course, hypothetically, I get a job in a metro city similar to Bangalore in terms of venues and population density etc., I could easily use Data Science to gather insights on the City without having to go to the location and check. The problem that Data Science can address is that we don't have to move an inch to gather insights. Data can be gathered with use of websites and tools and we can apply all the techniques taught in Data Science course to sufficiently apply machine learning techniques.

Problem Statement: - I am moving from Bangalore to Mumbai because of a new job. I have been living in Bangalore in a certain area for last few years and I have been comfortable to the neighborhood. I would like to find out if I can use the location data and check if I can find similar neighborhoods in Mumbai that could give me similar venues. How are the Mumbai locations placed in terms of availability of distinct venues compared to what we have in Bangalore? Can I shortlist the good neighborhood fit within a large metro as Mumbai?

Target Audience - This scenario helps anyone who is looking to move to a new city that is like the present city they live in. Or anyone who want to get an idea of a new city they want to move to. To find out how similar or dissimilar the new city is compared to the present city they are residing currently.

2. Data:- Data acquisition and cleaning

Data Sources:-

We will use foursquare data readily available to get data based on latitudes and longitudes as learnt in the tutorial earlier.

To address the Business problem, we would first need data of Bangalore and Mumbai locations. One way to gather data is based on zip codes. Since zip codes are unique and easily available from directly searching through google manually we can consider zip codes of Bangalore and Mumbai. Looking at the websites we can see both Bangalore and Mumbai have data available from website. There are other websites where a csv file is available but it was not easy to gather it for our project so we can skip that option. Hence every time we run the project from the notebook we need to connect to website and gather data. If the website is down or the information in the website changes then the results could change.

Data will be in tabular format and will have columns like City, Pin Code or Postal Code, Area. This will not be in ready to use format. Data Cleaning is necessary since there are certain zip codes that won't fetch latitude or longitude values easily and can create errors during program execution. Once the Data clean up happens, we can loop through and get coordinates and then we could proceed with using Foursquare.

Raw Data Extracted from the website could look like the sample data shown below: -

Area	**Postal Code**
Adugodi	560 030
Agaram	560 007
Air Force Stn. Yelahanka	560 063
Arabic College	560 045
Banashankari	560 050

We will use the Postal Code to get the Latitude and Longitude Values.

Use Pandas dataframe read_html method to read the data from the website. It return a list. The index 0 of the list contains the tabular data from the website.

Data Cleaning: -

The Postal Code obtained from the website has a space in between. This doesn’t help. We need to remove the space. Using the string function we can remove the space. Once we have the list of pin code with Area we need up update the column names. The tabular data that came from website has column header as 0 & 1. Both of these are integer values and do not help with our dataframe processing. So we need to convert them to string values and proper column names like zip_code and area. Next to get data from geopy library we have to use the Nominatim function and when we do that in a loop for each zip code we observe that some zip codes never return coordinate values. Observe such zip codes and we can either remove

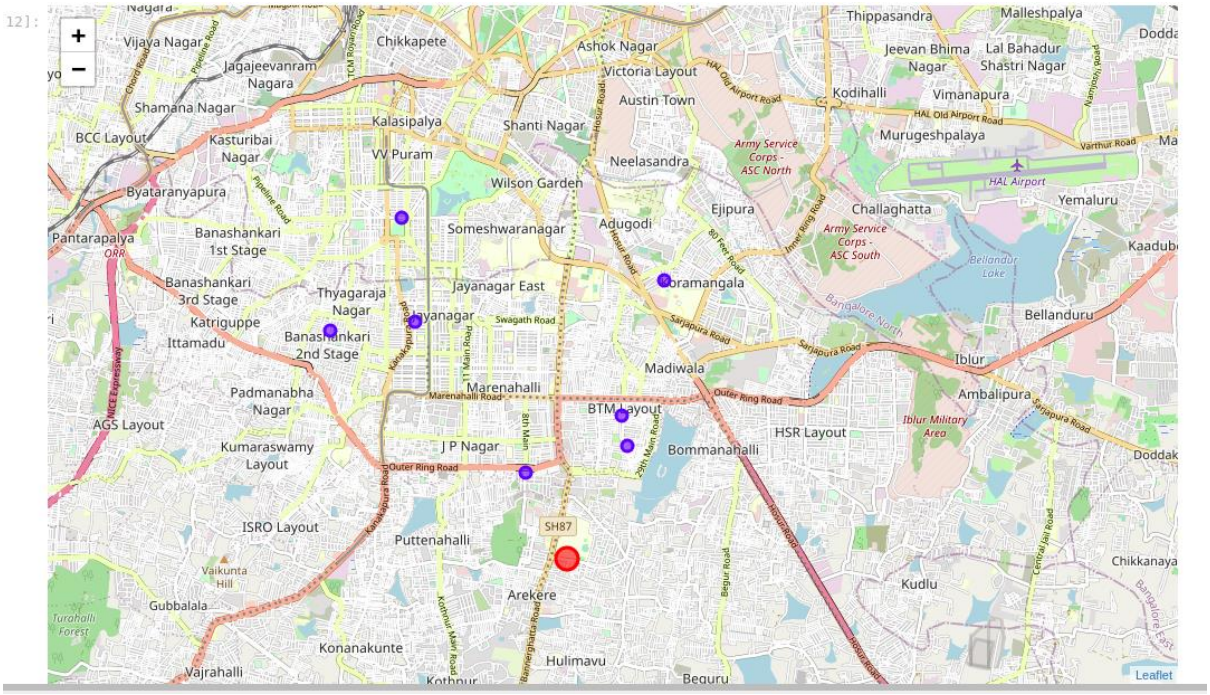
them from our dataset or manually find the coordinates for these areas and add them to our dataset. I have observed that we only have 3 such zip codes. Hence I decided to drop these by hard coding in the program.

Similar Data cleaning task was done for Mumbai zip code data. Mumbai zip code data had duplicates also. Since a zip code encompasses large area there were multiple areas within a single zip code. Hence the zip code values were not unique. I had to remove duplicates using dataframe function.

After all the clean up we now have final data ready for further processing. However, we see that there are too many zip codes to process. So I have filtered them to use a subset of data. For example: South bangalore locations are chosen as subset of data for bangalore data processing. Similarly thane and navi mumbai areas are used as subset for mumbai data processing.

3. Exploratory Data Analysis: -

Check the density of venues data given by FourSquare API.
Get nearby Italian restaurants surrounding an area of radius 3km from a given latitude and longitude coordinates. This latitude and longitude values are chosen based on where I live. So we see there are only 7 Italian restaurants within the given range.

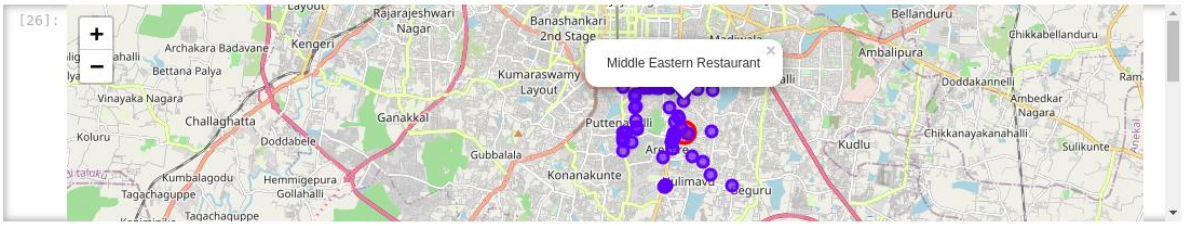


As we can see we have got very few restaurants within 5km range. It seems the data is less. It could be because there are more data points with Google places since Android phones are more widely used in India compared to Apple phones.

Checking venues in general instead of 'Italian' might fetch us more distinct venues. Let's explore other venues from the same latitude and longitude values as before.

```
# display map
venues_map
```

[26]:



This time around we have got many venues in groups and more data points in small space.

Observations

These are densely populated Areas. Even then, the venues list is less. I think this is because most people use android phone and since foursquare mainly collects data from iphone users the reviews and venue data is less. We can see some major grouping hotspots around 3-4 places. Areas like arekere gate, jayadeva flyover and 24th main road JP Nagar.

Many Gyms, tennis courts, sports places, swimming pools and many such places are not present. It could be due to how FourSquare returns the results as well.

Either that or there is a limit on how many venues we can query in a single call. Even though I have increased the radius and the limit I still get only 100 venues.

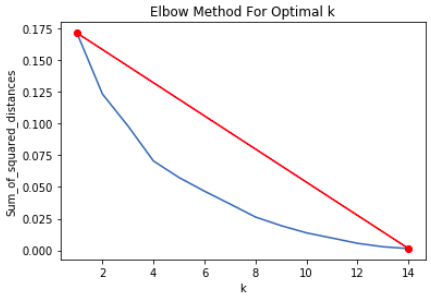
One more approach is to check data of Google places and see if we get more venues. More the data better the diversity of venues. I will leave that exercise to a later point in time.

4. K-means Clustering

K-means Clustering is possibly the best algorithm or model to be applied here. Venues returned from FourSquare API for each zip code act as input for K-means but not before we change the categorical variables into columns with binary values using onehot encoding. We need to find the optimum K for the k-means using the elbow method first.

I am trying to find out the optimum K value rather than assuming it to be 5 or 6 or 7

[43]:
```
Sum_of_squared_distances = []
bangalore_grouped_clustering = bangalore_grouped.drop('Neighborhood', 1)
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(bangalore_grouped_clustering)
    Sum_of_squared_distances.append(km.inertia_)

x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
import matplotlib.pyplot as plt
plt.plot(x, Sum_of_squared_distances)
plt.plot([K[0], K[13]],[Sum_of_squared_distances[0],Sum_of_squared_distances[13]],'ro-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
plt.show()
```



Since it is difficult to visually check if the optimum value is 4 or 5 lets use some mathematical function to check which point is farthest from the red line.

Let's confirm using a function

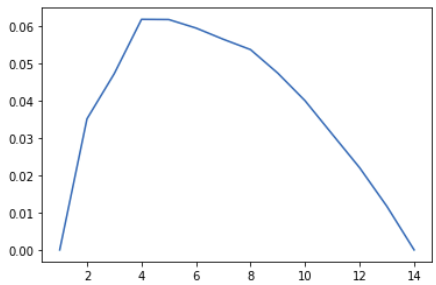
[44]:
```
def calc_distance(x1,y1,a,b,c):
    d = abs((a * x1 + b * y1 + c)) / (math.sqrt(a * a + b * b))
    return d
```

[45]:
```
a = Sum_of_squared_distances[0] - Sum_of_squared_distances[13]
b = K[13] - K[0]
c1 = K[0] * Sum_of_squared_distances[13]
c2 = K[13] * Sum_of_squared_distances[0]
c = c1 - c2
```

[46]:
```
import math
distance_of_points_from_line = []
for k in range(14):
    distance_of_points_from_line.append(
        calc_distance(K[k],Sum_of_squared_distances[k],a,b,c))
```

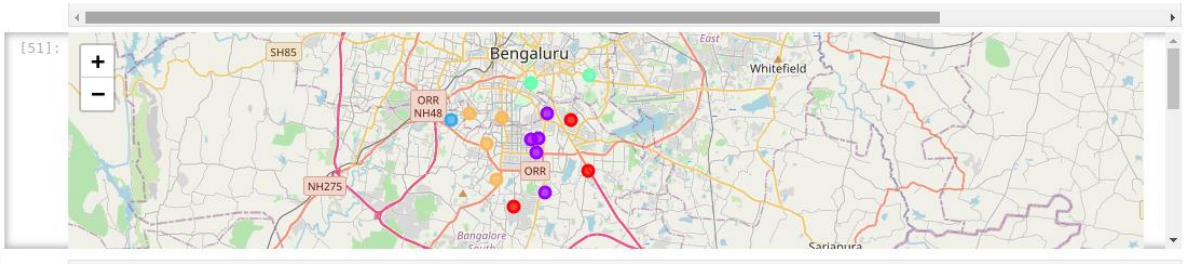
[47]:
```
distance_of_points_from_line
plt.plot(K,distance_of_points_from_line)
```

[47]:
```
[<matplotlib.lines.Line2D at 0x7f77f75e00d0>]
```



K=4 as can be seen from this plot. However, depending on what FourSquare returns we sometimes get k=5. I am considering k=5 since most times I get k=5.

Apply k-means clustering with k=5.



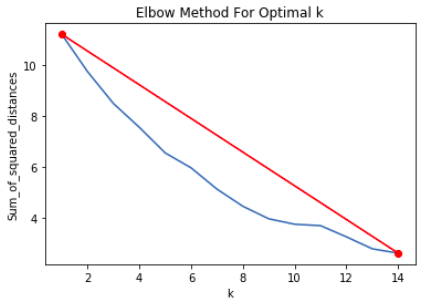
Observations: -

This has created 5 clusters and the observations I can take from this are: -
The cluster with most data points belong to the purple cluster [Cluster 1]. I stay in one of the purple clusters. One way to find results now is to find out cluster that has most data points for Mumbai data.

Most of the data are restaurants/Cafe. Although this is true globally, I feel we should have more data to have a diversity. Nonetheless let's go with what we have.

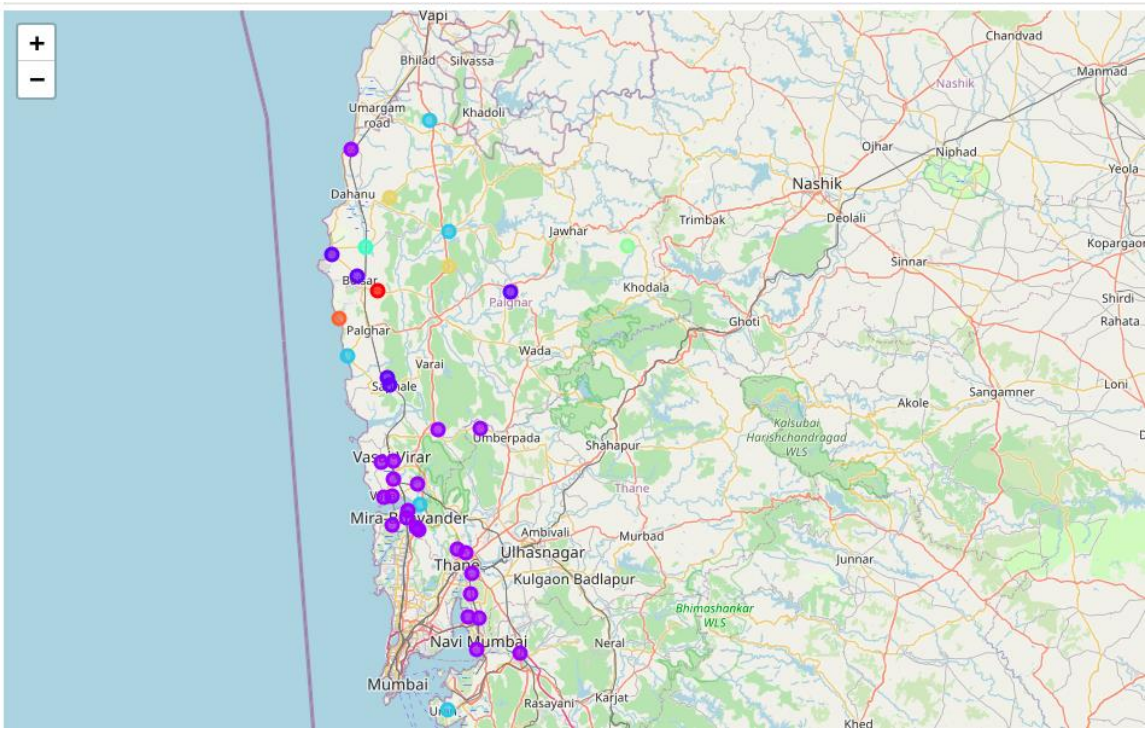
Plotting Optimum value of K for Mumbai data

[64]:
```
Sum_of_squared_distances = []
mumbai_grouped_clustering = mumbai_grouped.drop('Neighborhood', 1)
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(mumbai_grouped_clustering)
    Sum_of_squared_distances.append(km.inertia_)

x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
import matplotlib.pyplot as plt
plt.plot(x, Sum_of_squared_distances)
plt.plot([K[0], K[13]],[Sum_of_squared_distances[0],Sum_of_squared_distances[13]],'ro-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Let's plot the distance from the red line to see which K is optimum. K=8.
Fitting the model at k=8

[69]:



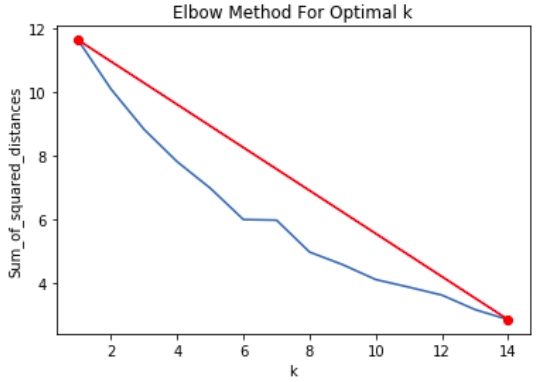
Again, the purple cluster seems to be very common with most data points.

If we inspect the cluster1 [purple] we get many areas. All of these seems to be similar in the sense they have restaurants and café to be the common places.

We can choose these areas for our consideration of areas similar to the current residing place in Bangalore.

However we can further explore with the data by merging both Bangalore and Mumbai data and find out the areas in Mumbai that fall in the same cluster as Bangalore cluster where I live.

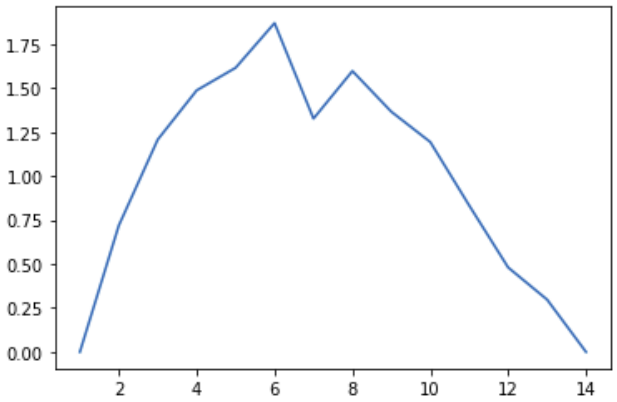
[79]:
```
Sum_of_squared_distances = []
all_grouped_clustering = all_grouped.drop('Neighborhood', 1)
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(all_grouped_clustering)
    Sum_of_squared_distances.append(km.inertia_)

x=[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
import matplotlib.pyplot as plt
plt.plot(x, Sum_of_squared_distances)
plt.plot([K[0], K[13]],[Sum_of_squared_distances[0],Sum_of_squared_distances[13]],'ro-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



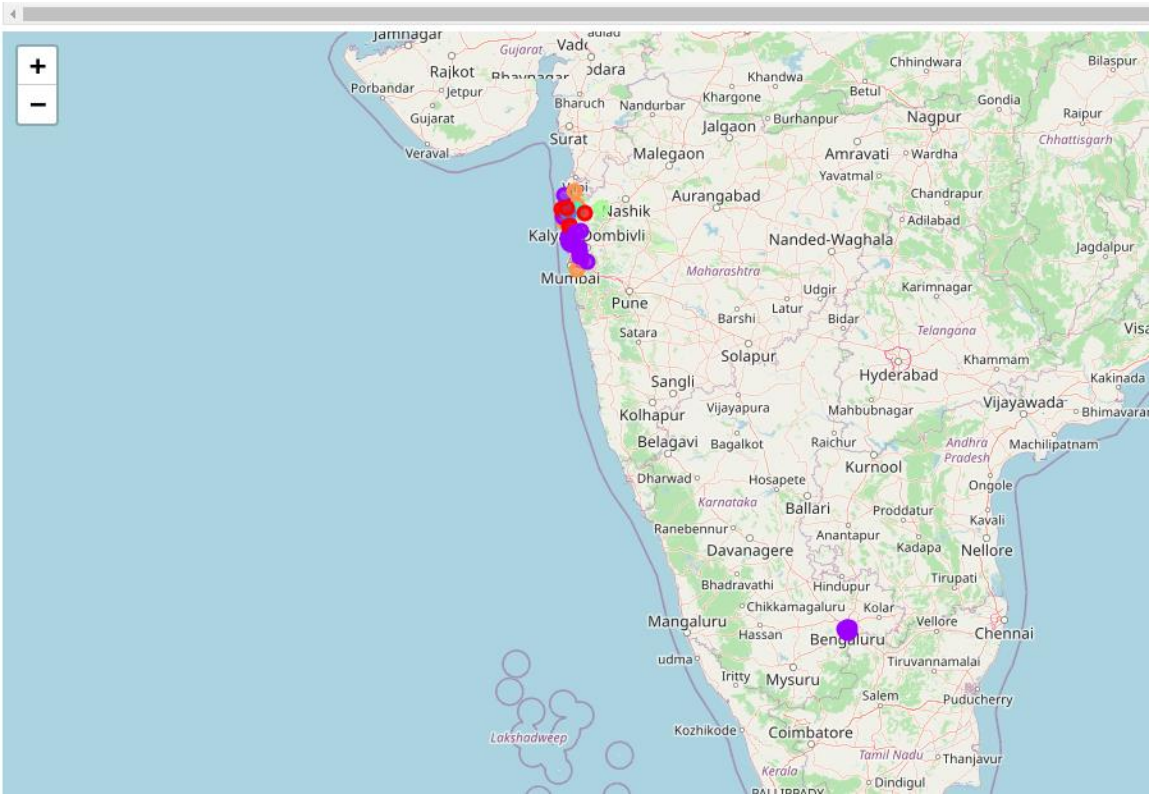
K=6.

[80]:
```
def calc_distance(x1,y1,a,b,c):
    d = abs((a * x1 + b * y1 + c)) / (math.sqrt(a * a + b * b))
    return d
a = Sum_of_squared_distances[0] - Sum_of_squared_distances[13]
b = K[13] - K[0]
c1 = K[0] * Sum_of_squared_distances[13]
c2 = K[13] * Sum_of_squared_distances[0]
c = c1 - c2

import math
distance_of_points_from_line = []
for k in range(14):
    distance_of_points_from_line.append(
        calc_distance(K[k],Sum_of_squared_distances[k],a,b,c))

distance_of_points_from_line
plt.plot(K,distance_of_points_from_line)
```

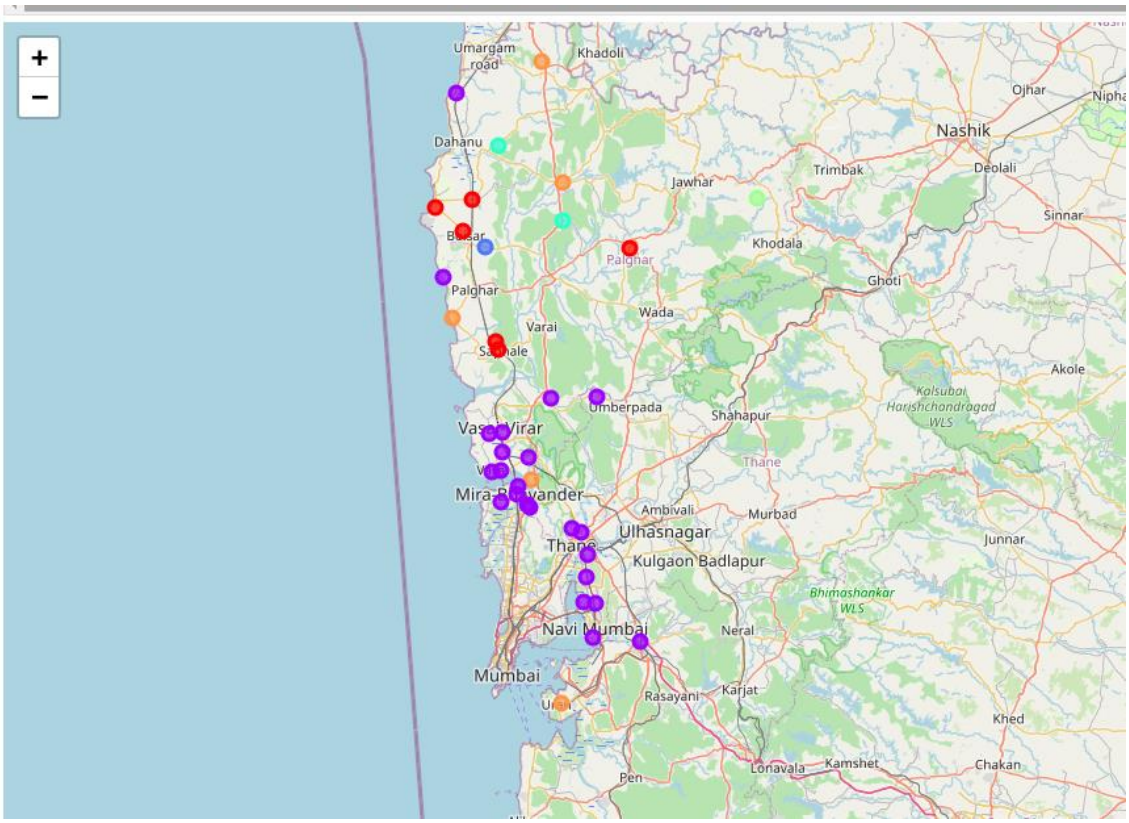
[80]:
```
[<matplotlib.lines.Line2D at 0x7f77f68ec210>]
```



[85]:


Jamnagar
Gujarat
Rajkot
Porbandar
Jetpur
Gujarat
Veraval
Bharuch
Nandurbar
Khargone
Khandwa
Chhindwara
Bilaspur
Betul
Gondia
Nagpur
Raipur
Chhattisgarh
Jalgaon
Burhanpur
Surat
Malegaon
Amravati
Wardha
Yavatmal
Aurangabad
Chandrapur
Adilabad
Nanded-Waghala
Jagdalpur
Maharashtra
Mumbai
Pune
Udgir
Karimnagar
Latur
Barshi
Bidar
Satara
Telangana
Solapur
Khammam
Hyderabad
Kakinada
Sangli
Vijayawada
Kolhapur
Vijayapura
Mahbubnagar
Belagavi
Bagalkot
Raichur
Andhra Pradesh
Machilipatnam
Kurnool
Dharwad
Hosapete
Ongole
Karnataka
Ballari
Proddatur
Kavali
Ranebennur
Anantapur
Kadapa
Nellore
Davanagere
Tirupati
Bhadravathi
Hindupur
Chikkamagaluru
Kolar
Vellore
Mangaluru
Hassan
Bengaluru
Chennai
udma
Tiruvannamalai
Iritty
Mysuru
Salem
Puducherry
Lakshadweep
Kozhikode
Coimbatore
Tamil Nadu
Thanjavur
Kerala
Dindigul

[85]:


Umargam
road
Khadoli
Ojhar
Nipha
Nashik
Dahanu
Trimbak
Deolali
Jawhar
Sinnar
Khodala
Palghar
Ghoti
Palghar
Wada
Varai
Akole
Kolsubai
Harishchandragad
WLS
Umberpada
Shahapur
Thane
Ambivali
Murbad
Ulhasnagar
Junnar
Kulgaon Badlapur
Bhimashankar
WLS
Navi Mumbai
Neral
Mumbai
Rasayani
Karjat
Khed
Pen
Lonavala
Kamshet
Chakan

5. Conclusions: -

Cluster purple (cluster 1) seems to be with most data points and it is the same cluster that contains the bangalore bannerghatta road zip code area.

That settles it. I can now consider the purple cluster data points to be similar in nature with venues and frequency.

6. Future Directions: -

Although our analysis seems to be OK, I would be more comfortable with more data points from either foursquare or switching to Google places. Maybe next time!