Coursera, Applied Data Science Capstone :- The Battle of Neighborhood

---

The-Battle-of-Neighborhoods

Problem 1.
Which city amongst these is best to stay if you have to survive only on pizza?

Problem 2.
Which city amongst these should one consider if you want to setup a pizza outlet.

Introduction
Have taken few major Indian cities like Mumbai, Delhi, Kolkata, Pune & Bengaluru to do the above analysis. Suppose if you have to visit one of the cities then which city would you go to where you get pizza easily and one don't need to travel far for the pizza. Solving the first scenario we get to solve one more problem that's if say suppose one wants to invest or start a pizza outlet then one should invest in which cities from the above selected cities?

So it for people who want to go to a place with a high density of Pizza around them. The problem we aim to solve is to analyze the Pizza stores' locations in these Indian cities and find the best place for our tourist so that he can have a good time enjoying the piza. Our main target are tourists with a taste of western-style pizza. Second is for those peoples who are looking for the best cities to start piza outlets.

So it for people who want to go to a place with a high density of Pizza around them. The problem we aim to solve is to analyze the Pizza stores' locations in these Indian cities and find the best place for our tourist so that he can have a good time enjoying the pizza. Our main target are tourists with a taste of western-style pizza. Second is for those peoples who are looking for the best cities to start pizza outlets.

Will use the FourSquare API to collect data about locations of Pizza stores in major Indian cities which are: Mumbai, Delhi, Kolkata, Pune & Bengaluru, these are few of the most populated Indian cities and I am hopeful that they will contain the best Pizza places in that cities. Using the same Api and dataset we will also try to figure out which city has lesser density of pizza outlets , making way for setting up new pizza outlet opportunities.

Methodology & Data Gathering

The main target here is to asses which city would have the highest Pizza store density. I used the Four Square API through the venue channel. I used the near query to get venues in the cities. Also, I use the CategoryID to set it to show only Pizza Places. API Request format for Venue :
.e.g.https://api.foursquare.com/v2/venues/explore?&client_id=&client_secret=&v=20180605&Mumbaik,Delhi&limit=100&categoryId=4bf58dd8d48988d1ca941735 (https://api.foursquare.com/v2/venues/explore?&client_id=&client_secret=&v=20180605&Mumbaik,Delhi&limit=100&categoryId=4bf58dd8d48988d1ca941735)
Also, Foursquare limits us to maximum of 100 venues per query. Moreover, I repeated this request for the 5 studied cities and got their top 100 venues. I saved the name and coordinate data only from the result and plotted them on the map for visual inspection.Next, to get an indicator of the density of Pizza Places, I calculated a center coordinate of the venues to get the mean longitude and latitude values. Then calculated the mean of the Euclidean distance from each venue to the mean coordinates. That was my indicator; mean distance to the mean coordinate. Using these methodology, we will be soloving the above two solution.

Key Datasource : Foursquare API's

Core coding & libraries section

Step1

Import and instalation of key libraries ,PANDAS & FOLIUM

In [31]:

```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas d
ataframe
#!conda install -c conda-forge folium=0.5.0 --yes # Install folium
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

Step 2

Setup of Foursquare API using credentails

In [2]:
```
CLIENT_ID = 'foursquare client id' # your Foursquare ID
CLIENT_SECRET = 'foursquare id' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
```

```
Your credentails:
CLIENT_ID: As per foursquare
```

Step 3

Calling the FS API and passing the venues parameter to fetch data for pizza outlets for the said cities ie. 'Mumbai', 'Delhi', 'Kolkata', 'Pune' & 'Bengaluru'

In [4]:
```
LIMIT = 200 # Maximum is 100
#cities = ["New York, NY", 'Chicago, IL', 'San Francisco, CA', 'Jersey City, N
J', 'Boston, MA']
cities = ['Mumbai', 'Delhi', 'Kolkata', 'Pune', 'Bengaluru']
results = {}
for city in cities:
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_s
ecret={}&v={}&near={}&limit={}&categoryId={}'.format(
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        city,
        LIMIT,
        "4bf58dd8d48988d1ca941735") # PIZZA PLACE CATEGORY ID
    results[city] = requests.get(url).json()
```

In [5]:
```
df_venues={}
for city in cities:
    venues = json_normalize(results[city]['response']['groups'][0]['items'])
    df_venues[city] = venues[['venue.name', 'venue.location.address', 'venue.l
ocation.lat', 'venue.location.lng']]
    df_venues[city].columns = ['Name', 'Address', 'Lat', 'Lng']
```

Note : The Foursquare API Only gives us the nearest 100 venues in the city.

Let's first list them out and check for there densities.

In [32]:
```
maps = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['boun
ds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['boun
ds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['boun
ds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['boun
ds']['sw']['lng']])
    maps[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)

    # add markers to map
    for lat, lng, label in zip(df_venues[city]['Lat'], df_venues[city]['Lng'],
df_venues[city]['Name']):
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='#3186cc',
            fill_opacity=0.7,
            parse_html=False).add_to(maps[city])
    print(f"Total number of pizza outlets in {city} = ", results[city]['respon
se']['totalResults'])
    #print("Showing Top 100")
```

```
Total number of pizza outlets in Mumbai =  127
Total number of pizza outlets in Delhi =  36
Total number of pizza outlets in Kolkata =  25
Total number of pizza outlets in Pune =  89
Total number of pizza outlets in Bengaluru =  117
```

Display of results on map

Step 4

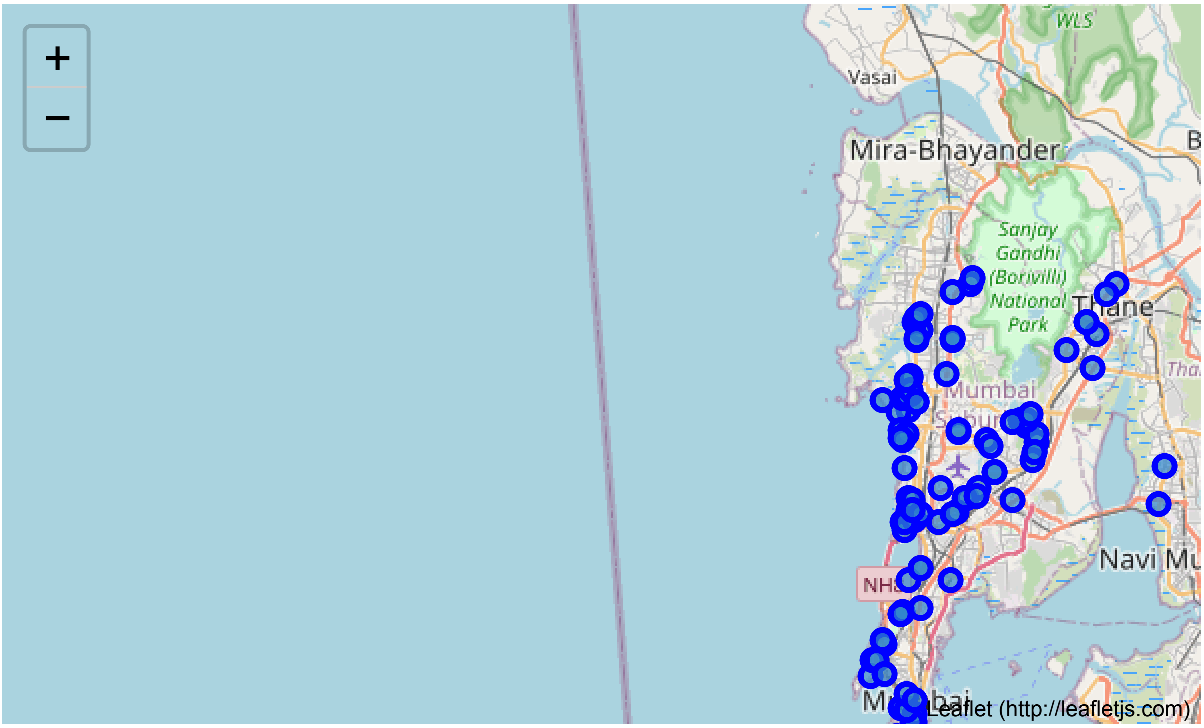
Data analysis & visulaziation using foluim.

Lets plot these on Map.

Ploting outlets on Map city wise as per the array [0] 'Mumbai' [1] 'Delhi'[2] 'Kolkata'[3] 'Pune' [4] 'Bengaluru'

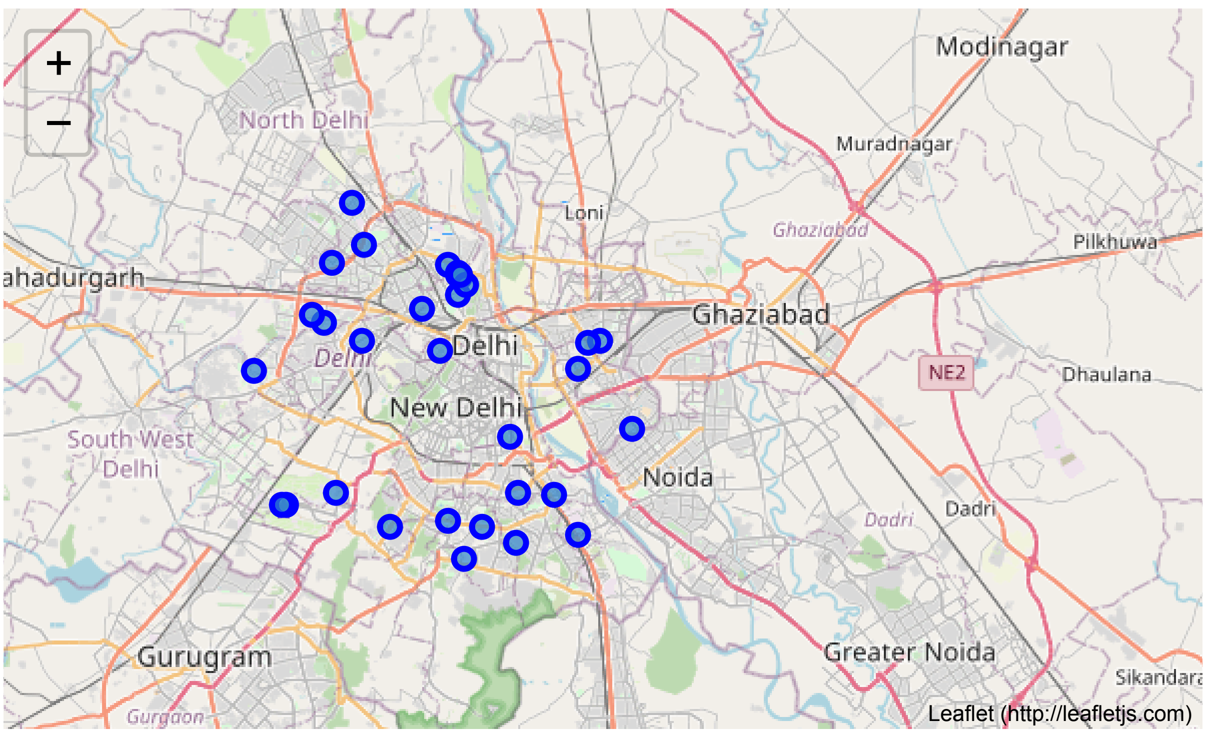
In [33]:
```
#City Mumbai Array of 0
maps[cities[0]]
```

Out[33]:



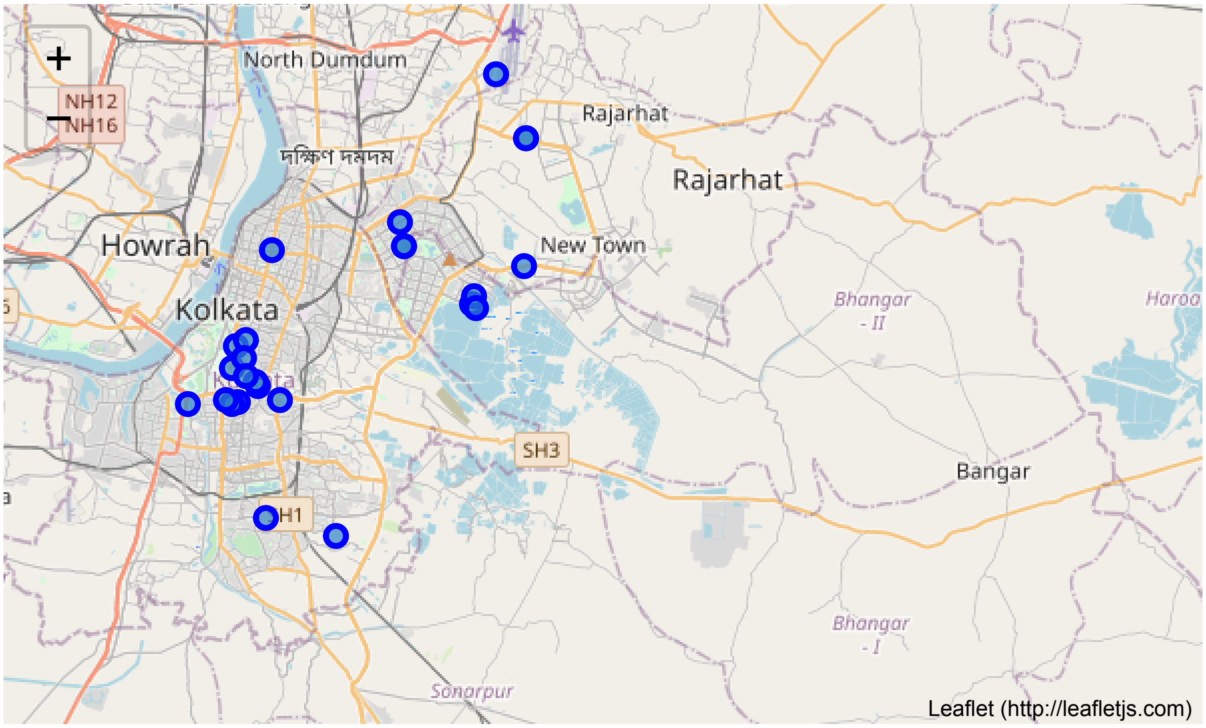
In [34]:
```
#City Delhi Array of 1
maps[cities[1]]
```

Out[34]:



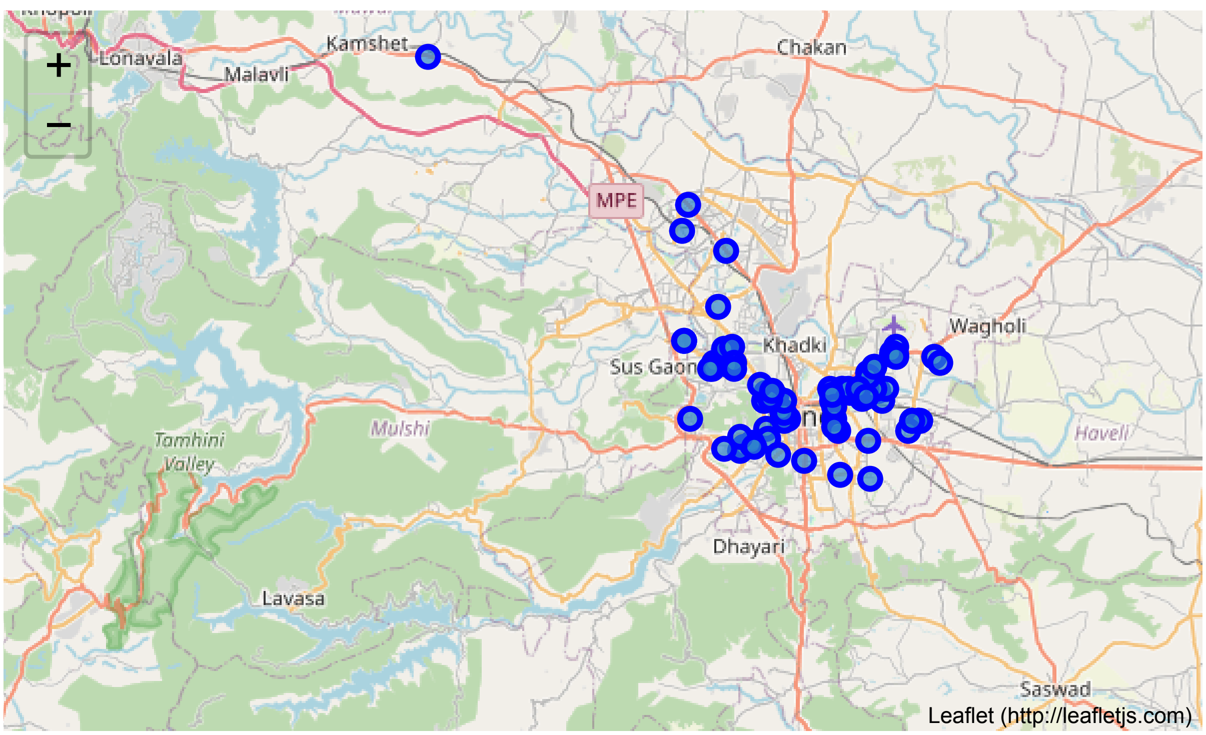
In [12]:
```
#City Kolkata Array of 2
maps[cities[2]]
```

Out[12]:



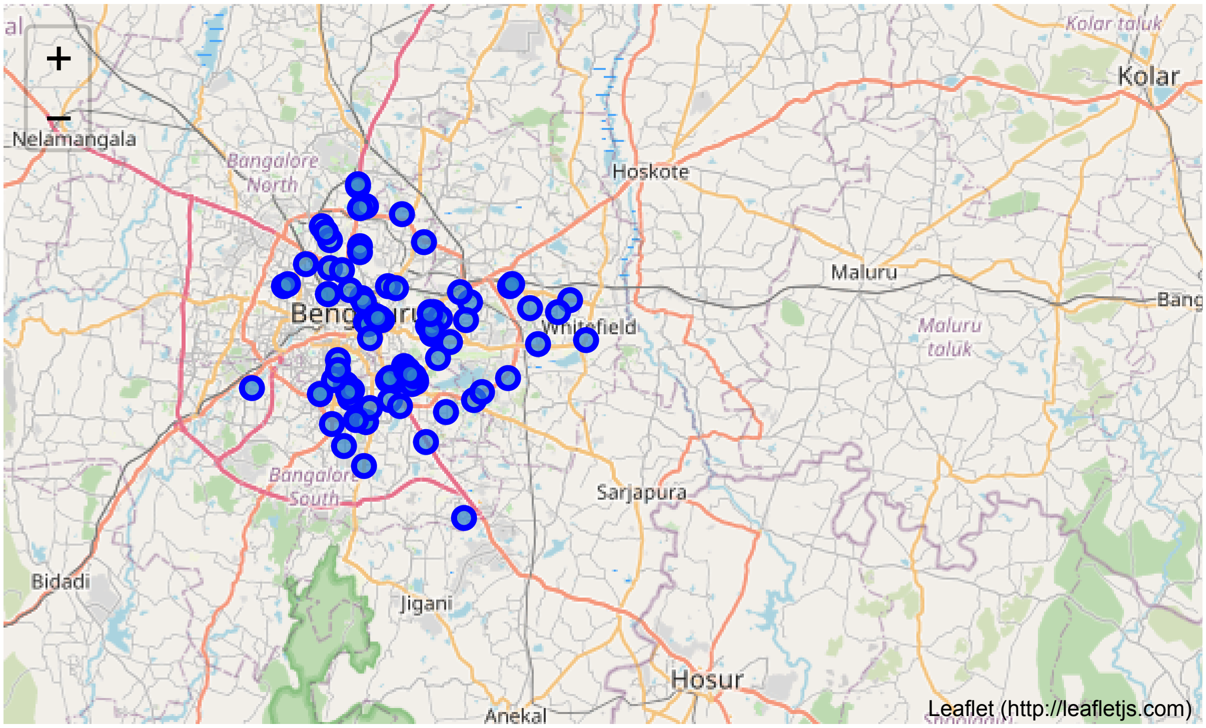
In [35]:
```
#City Pune Array of 3
maps[cities[3]]
```

Out[35]:



```
In [36]: #City Bengaluru Array of 4
         maps[cities[4]]
```

Out[36]:




```
In [ ]:
```

As per the count
Total number of pizza outlets in Mumbai = 127
Total number of pizza outlets in Delhi = 36
Total number of pizza outlets in Kolkata = 25
Total number of pizza outlets in Pune = 89
Total number of pizza outlets in Bengaluru = 117

We can see that Mumbai and Bengaluru are the most dense cities with Pizza outlets. However, Let's have a concrete measure of this density.
For this I will use some basic statistics. I will get the mean location of the pizza places which should be near to most of them if they are really dense or far if not.
Next will take the average of the distance of the venues to the mean coordinates to calculate this.

Calculating & displaying of the mean coordinates

Step 5

Calculating mean coordinates

In [37]:

```
maps = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['boun
ds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['boun
ds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['boun
ds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['boun
ds']['sw']['lng']])
    maps[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)
    venues_mean_coor = [df_venues[city]['Lat'].mean(), df_venues[city]['Lng'].
mean()]
    # add markers to map
    for lat, lng, label in zip(df_venues[city]['Lat'], df_venues[city]['Lng'],
df_venues[city]['Name']):
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='#3186cc',
            fill_opacity=0.7,
            parse_html=False).add_to(maps[city])
        folium.PolyLine([venues_mean_coor, [lat, lng]], color="green", weight=
1.5, opacity=0.5).add_to(maps[city])

    label = folium.Popup("Mean Co-ordinate", parse_html=True)
    folium.CircleMarker(
        venues_mean_coor,
        radius=10,
        popup=label,
        color='green',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(maps[city])

    print(city)
    print("Mean Distance from Mean coordinates")
    print(np.mean(np.apply_along_axis(lambda x: np.linalg.norm(x - venues_mean
_coor),1,df_venues[city][['Lat','Lng']].values)))
```

```
Mumbai
Mean Distance from Mean coordinates
0.08227507521263808
Delhi
Mean Distance from Mean coordinates
0.0901479755022745
Kolkata
Mean Distance from Mean coordinates
0.048636923957809934
Pune
Mean Distance from Mean coordinates
0.05564469825150924
Bengaluru
Mean Distance from Mean coordinates
0.05526520972026041
```

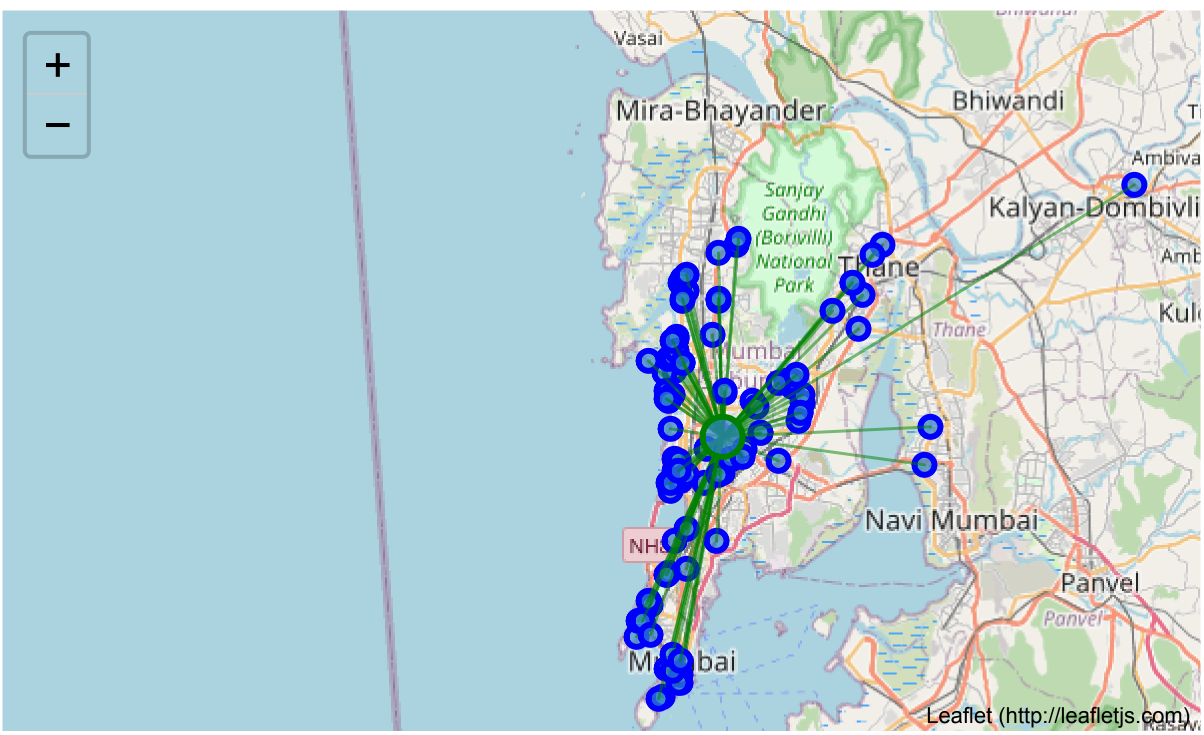
Ploting of the mean coordinates

Step 6

In this phase we Calculate the Mean coordinate and the mean distance to mean coordinate(MDMC).
Lets plot these on Map.
Ploting outlets on Map city wise as per the array [0] 'Mumbai' [1] 'Delhi'[2] 'Kolkata'[3] 'Pune' [4] 'Bengaluru'
We represent the mean coordinate with a big green circle and distances with green lines

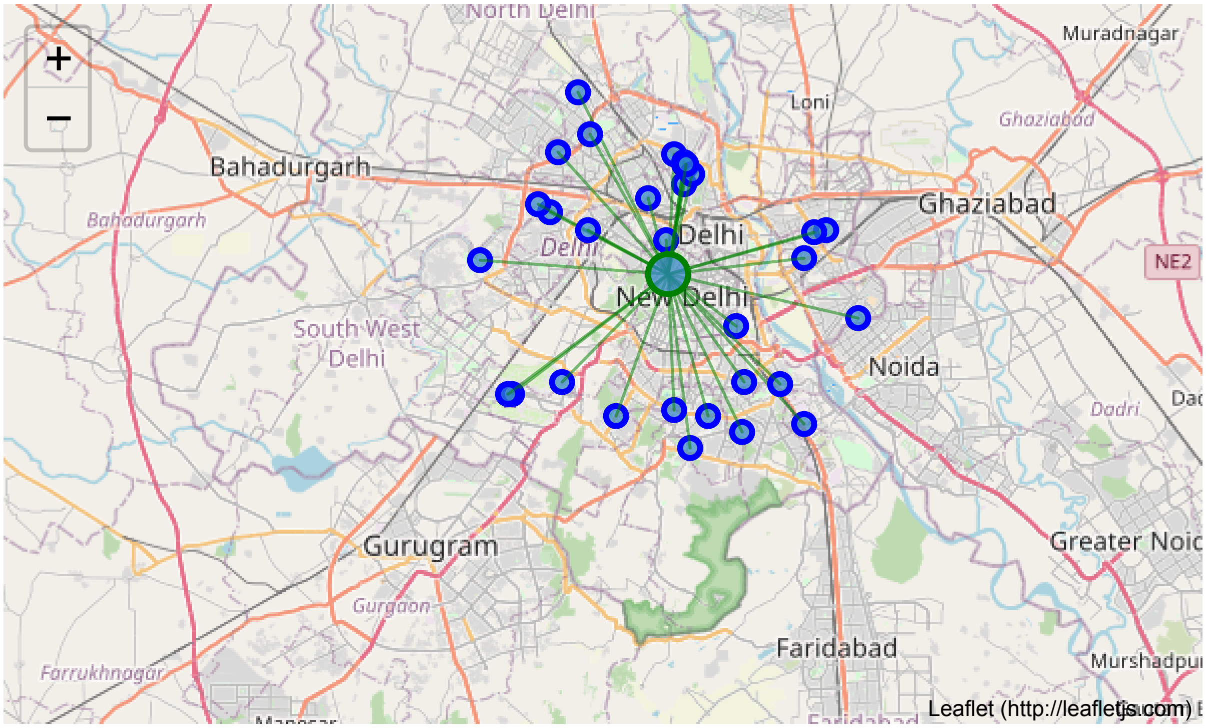
In [22]:
```
#City Mumbai Array of 0
maps[cities[0]]
```

Out[22]:



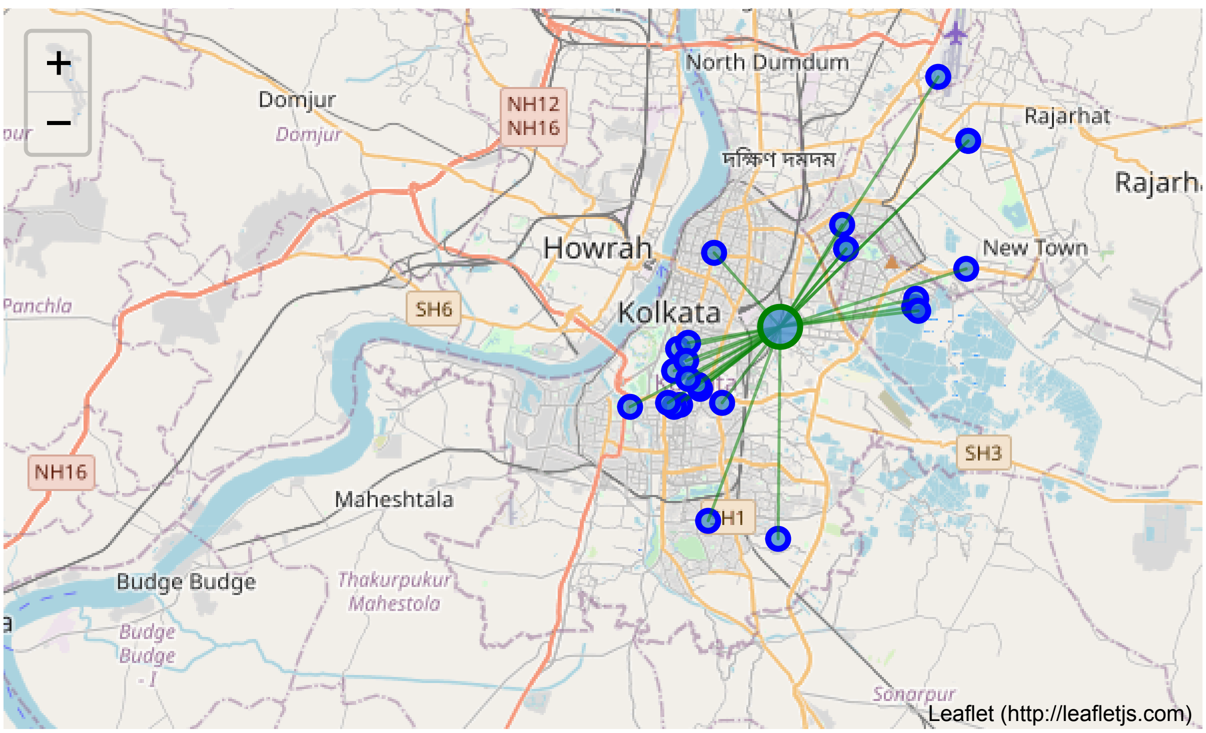
In [23]:
```
#City Delhi Array of 1
maps[cities[1]]
```

Out[23]:



In [24]:
```
#City Kolkata Array of 2
maps[cities[2]]
```

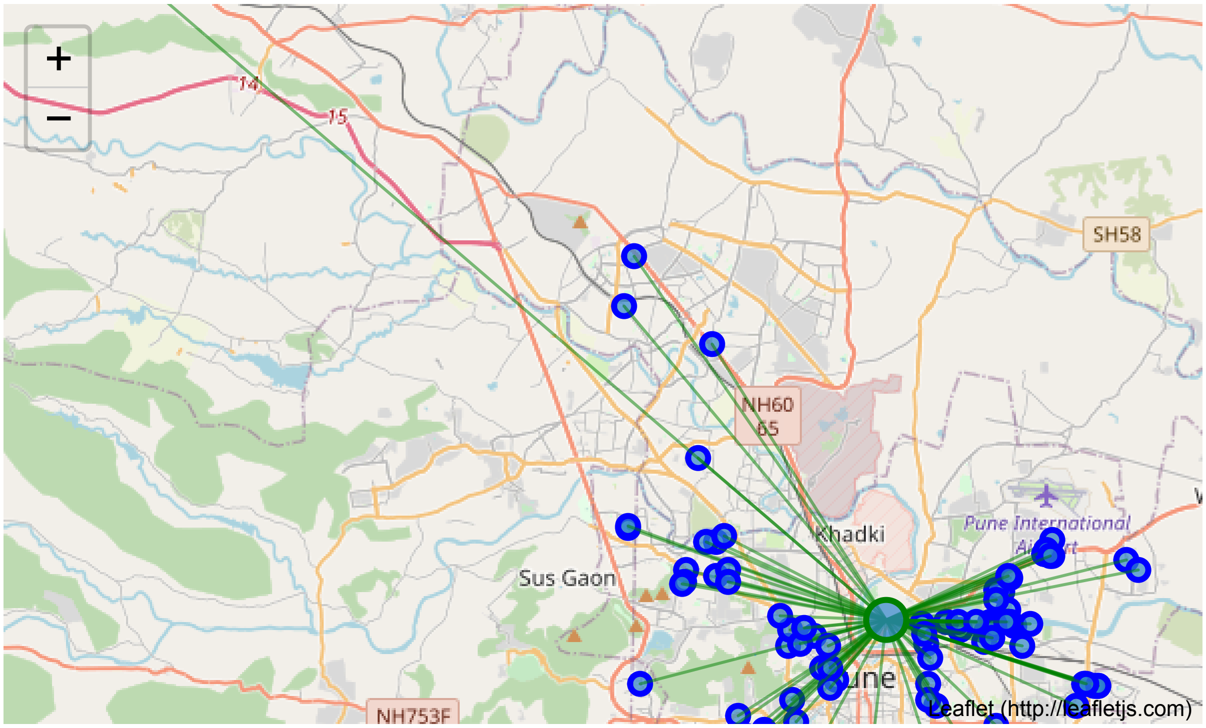
Out[24]:



In [25]:

```
#City Pune Array of 3
maps[cities[3]]
```

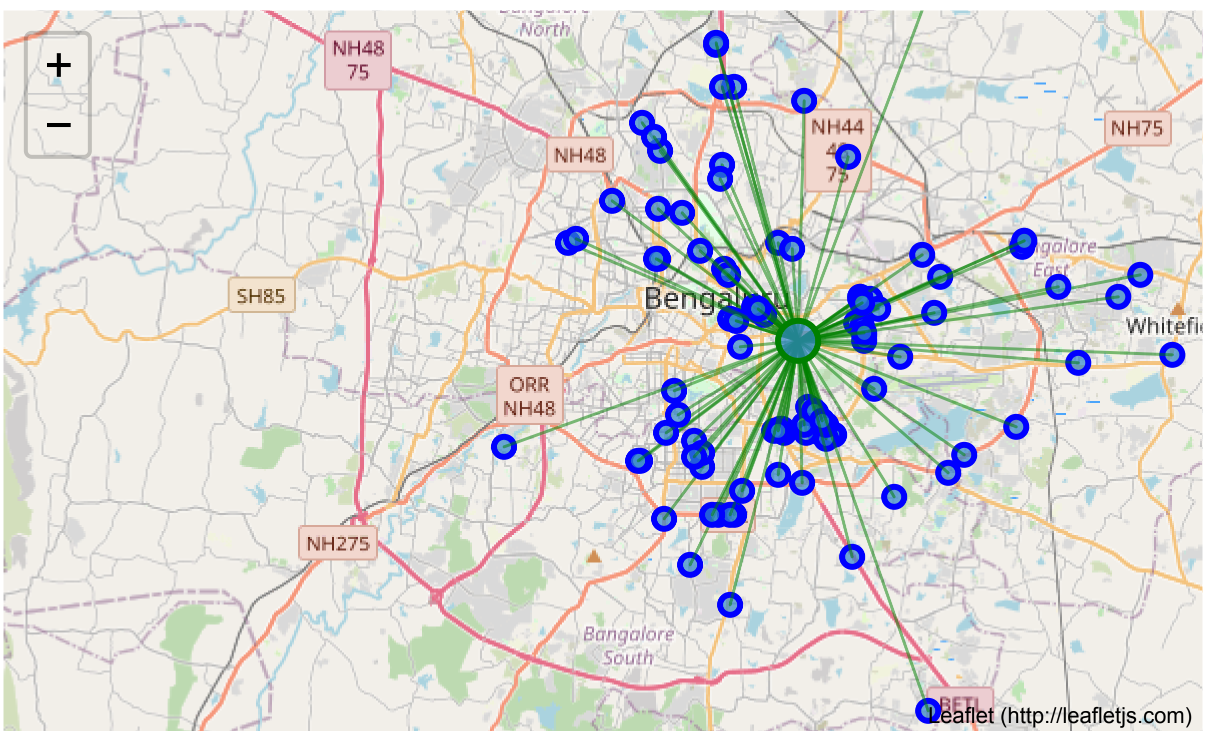
Out[25]:



In [26]:

```
#City Bengaluru Array of 4
maps[cities[4]]
```

Out[26]:



In [ ]:

Mumbai Mean Distance from Mean coordinates : 0.08227507521263808
Delhi Mean Distance from Mean coordinates : 0.0901479755022745
Kolkata Mean Distance from Mean coordinates : 0.048636923957809934
Pune Mean Distance from Mean coordinates : 0.05564469825150924
Bengaluru Mean Distance from Mean coordinates : 0.05526520972026041
We now see that Kolkata is the best option to stay if you need piza followed by Pune & Bengaluru. The best place to invest in an outlet is Mumbai & Delhi.Tourist's best interest would be to book a hotel near that mean coordinate to surround himself with the 100 Pizza stores there!!
Invester best choice would be Mumbai & Delhi

Discussion

The MDMC of 0.0486369 is the best for Kolkata.We see that Kolkata is the best option to stay if you need piza followed by Pune & Bengaluru. The best place to invest in an outlet is Mumbai & Delhi.Tourist's best interest would be to book a hotel near that mean coordinate to surround himself with the 100 Pizza stores there!!
Invester best choice would be Mumbai & Delhi

Conclusion

Best place where you get easy acess to pizza oulets: Kolkata followed by Pune & Bengaluru
Best place to invest in opening a new pizza outlet: Delhi & Mumbai

THANK YOU!!

Coursera IBM Applied Data Science Capstone :Capstone Project - The Battle of Neighborhoods (Week 2)