A Program for visitors map

INTRODUCTION

AGD travels is a company based in bangalore which provides digital maps for visitors which conatins information about the several neighbourhoods of the city. the information contains places to get food(including restaurants, cafes, hookah bars, bars ,etc.),landscapes, recreation spots, police stations and hospitals(for emergency). They also provide a cluster map which shows the neighbourhoods with similar food locations for customers with a knack of food blogging

DATA DESCRIPTION

The preliminary dataset is made from web scraping from a wikipedia page(https://en.wikipedia.org/wiki/List_of_neighbourhoods_in_Bangalore). The location coordinates are searched using the geopy library of the python language. The data used in this project is provided by Foursquare location data. Specific API calls are made for specific types of category. The category Id is used to get the various categorical venues. The data are grouped by neighbourhood areas, and each area includes the information about restaurants, cafes, landscapes, recreation spots, police stations and hospitals present in those area.

The web scraping is done using pythons library BeautifulSoup

```
import bs4
```

The Data collected from the website contains the names of all major neighbourhoods of the city of Bangalore.

Now the next step is to get the data of their location coordinates from the geopy library.

```
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
```

This data is then added to the main Data Frame containing the name of the neighbourhoods, The Data Set now looks something like this.

	Neighbourhood	Latitude	Longitude
0	Cantonment area	13.0196	77.5096
1	Domlur	12.9625	77.6382
2	Indiranagar	12.9733	77.6405
3	Jeevanbheemanagar	NaN	NaN
4	Malleswaram	13.0163	77.5587
5	Pete area	NaN	NaN
6	Sadashivanagar	13.0074	77.5794
7	Seshadripuram	12.9932	77.5753
8	Shivajinagar	12.9864	77.6075

As it is seen this Data needs refinement and additional fields to give meaningful results and fulfil the demands of the clients(as in our case are the visitors).

Methodology

The data collected is cleaned.
On Specific investigation of areas whose location data was not available it is found that such areas have no specific tourist attraction and our mainly industrial areas so can be removed from the dataset as visitors will not be interested in such areas.
This starts the data cleaning process.
All the data whose location coordinates are equal to **NaN** removed.

Now the next major step is to use the foursquare API calls to get the data of the major Places in the neighbourhoods.

First:

The location of the places which interest The Foodie inside any visitor.
Such places include restaurants, café, Bars, BBQ joints, Food Carts etc.

The data is received in Json format which is the processed and tabulated in the form of a Data Frame.

The following function is used to get the json using foursquare API calls and the tabulating them in the form of a Data frame:

```
def getNearbyVenues(names, latitudes, longitudes,categoryID,radius=500)
:

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id=
{}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&categoryId={}'.for
mat(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT,categoryID)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['it
ems']
        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name'],
            ) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list fo
r item in venue_list])
```

```
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

The data set looks like:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Cantonment area	13.019567	77.509589	New Udupi Grand	13.022775	77.509830	Vegetarian / Vegan Restaurant
1	Domlur	12.962467	77.638196	Lavonne	12.963909	77.638579	Café
2	Domlur	12.962467	77.638196	Barbeque Nation	12.962684	77.641599	BBQ Joint
3	Domlur	12.962467	77.638196	Smoke House Deli	12.965584	77.641498	Deli / Bodega
4	Domlur	12.962467	77.638196	Domino's Pizza	12.961000	77.639000	Pizza Place
5	Domlur	12.962467	77.638196	The Woodstok	12.966818	77.637182	Steakhouse
6	Domlur	12.962467	77.638196	Mainland China	12.962458	77.641727	Chinese Restaurant

The fields are :
1. Name of neighbourhoods: Neighborhood.
2. The Latitude of the Neighbourhood: Neighborhood Latitude
3. The Longitude of the Neighborhood: Neighborhood Longitude
4. The Name of the Venue: Venue
5. The Latitude of the Venue: Venue Latitude
6. The Longitude of the Venue: Venue Longitude
7. The Venue Category: Venue Category

This same process is used to get data of outdoor recreation places, Hospitals, Police stations etc.

Now as the data is received it is important to create pictorial representations of these places because the data itself is vast enough to confuse the visitor.
The best pictorial representation would be imposing this data on a map of these locations.

Maps are generated using Folium library of Python.

```
import folium
```

A basic function is used to get the map of the locations.
The need of this function is to reduce the task of recreating the code for generation of the map of Hospitals and police stations. This same code is capable to do it for every dataset containing the same field as mentioned above.

```
def gen_map(df):
    df_map = folium.Map(location=[12.9716, 77.5946], zoom_start=12)

    for lat, lng, name, categories in zip(df['Venue Latitude'], df['Ven
ue Longitude'],
                                          df['Venue'], df['Venue C
ategory']):

        label = '{}'.format(name)
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
```

```
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='blue',
            fill_opacity=0.7,
            parse_html=False).add_to(df_map)
    m=df_map

    return m
```

Now it is important for the foodie traveller to Know which neighbourhoods have similar food related places so that the visitor is able to plan important places to visit if they are in a time crunch.

This is done using K-means clustering algorithm.

The clusters are made on the basis of the top ten most common venues among the neighbourhoods.

For this purpose it is important to know which are the most common venues.
This is done by using the **Venue Category**.
Dummies are created for the values of different categories listed in the above mentioned field.
It looks like this:

	Neighborhood	American Restaurant	Andhra Restaurant	Asian Restaurant	BBQ Joint	Bakery	Bengali Restaurant	Bistro	Breakfast Spot	Burger Joint	Cafeteria	Café	Chaat Place	Chettinad Restaurant	Chinese Restaurant	Bc
0	Cantonment area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Domlur	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
2	Domlur	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
3	Domlur	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	Domlur	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Now the most common places are found using the following Function:

```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

and this code:

```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicator
s[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
```

```
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = food_grouped['Neighborhood']

for ind in np.arange(food_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(food_grouped.iloc[ind, :], num_top_venues)
```

The Table looks likes this:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Arekere	Indian Restaurant	Bakery	Mughlai Restaurant	Pizza Place	Breakfast Spot	Chinese Restaurant	Bengali Restaurant	Bistro	Hyderabadi Restaurant	Gastropub
1	BTM Layout	Indian Restaurant	Café	Snack Place	Vegetarian / Vegan Restaurant	Bakery	Chinese Restaurant	Sandwich Place	Restaurant	Dim Sum Restaurant	Diner
2	Banashankari	Indian Restaurant	Food Truck	Southern / Soul Food Restaurant	Pizza Place	Chinese Restaurant	Hyderabadi Restaurant	Gastropub	Food Court	Fast Food Restaurant	Eastern European Restaurant
3	Banaswadi	Indian Restaurant	Vegetarian / Vegan Restaurant	BBQ Joint	Bakery	Kerala Restaurant	Deli / Bodega	Hyderabadi Restaurant	Gastropub	Food Truck	Food Court
4	Basavanagudi	Indian Restaurant	Restaurant	Café	Snack Place	Mediterranean Restaurant	Indian Sweet Shop	Bengali Restaurant	Bistro	Gastropub	Food Truck

Now the K-means algorithm is used to find the clusters:

```
# set number of clusters
kclusters = 5

food_grouped_clustering = food_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(food_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

The kmeans.labels_ list store the value of the cluster to which each neighbourhood belongs:
The Cluster labels are then added to the above shown Dataset and the final dataset is formed.

	Neighbourhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue
0	Cantonment area	13.0196	77.5096	1.0	Vegetarian / Vegan Restaurant	Wings Joint	Chettinad Restaurant	Hyderabadi Restaurant	Gastropub	Food Truck	Food Court	Fast Food Restaurant	Eastern European Restaurant
1	Domlur	12.9625	77.6382	3.0	Indian Restaurant	Café	Sandwich Place	Vietnamese Restaurant	Italian Restaurant	Deli / Bodega	Chinese Restaurant	Pizza Place	Rajasthani Restaurant
2	Indiranagar	12.9733	77.6405	3.0	Indian Restaurant	Café	Restaurant	Italian Restaurant	Steakhouse	Bakery	Bistro	Chinese Restaurant	Diner
3	Malleswaram	13.0163	77.5587	1.0	Vegetarian / Vegan Restaurant	Wings Joint	Chettinad Restaurant	Hyderabadi Restaurant	Gastropub	Food Truck	Food Court	Fast Food Restaurant	Eastern European Restaurant
4	Sadashivanagar	13.0074	77.5794	3.0	Indian Restaurant	Café	Vegetarian / Vegan Restaurant	Snack Place	Fast Food Restaurant	Chinese Restaurant	Seafood Restaurant	Pizza Place	Bakery

Folium is again used to create a map showing each neighbourhood coloured in different colours belonging to each cluster. Neighbourhoods of the same colours are similar in Food related places.

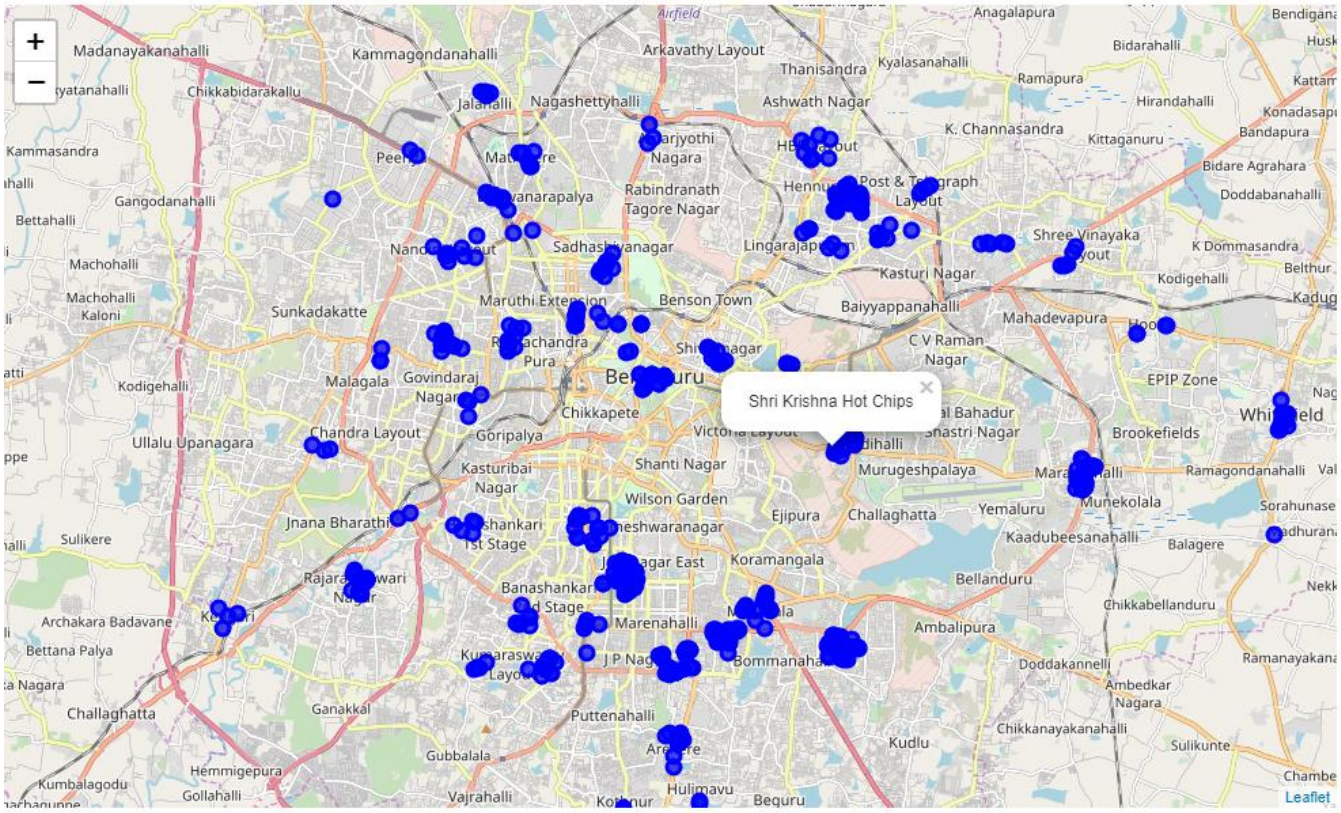
```
# create map
map_clusters = folium.Map(location=[12.9716, 77.5946], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
```

```
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(food_merged['Latitude'], food_merged[
'Longitude'], food_merged['Neighbourhood'], food_merged['Cluster Labels
']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_h
tml=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

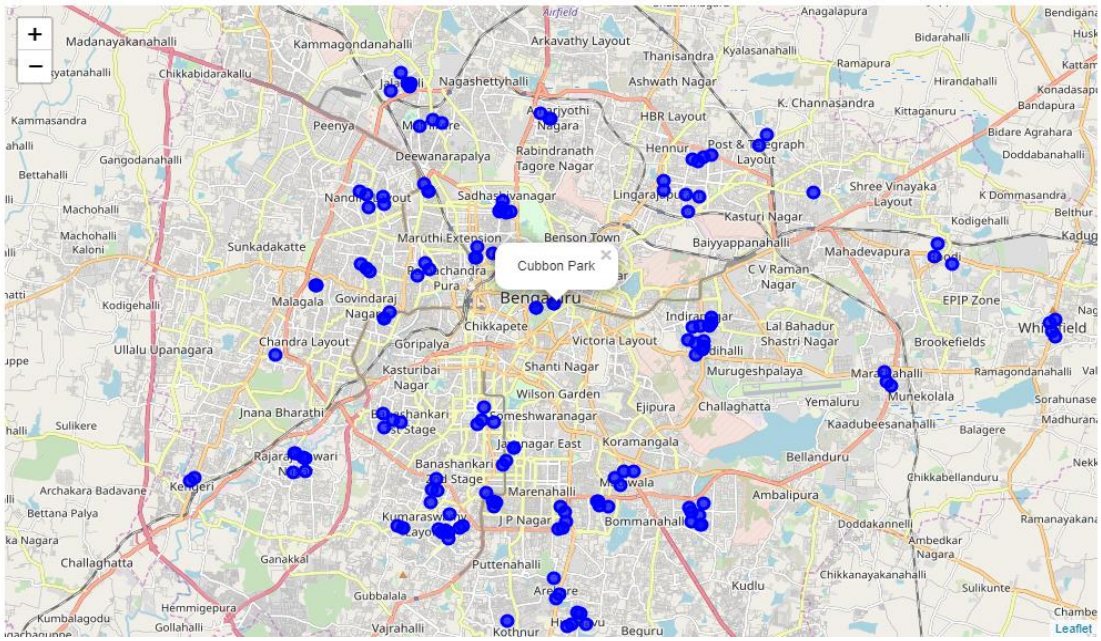
Results

The Maps Generated are as follows:

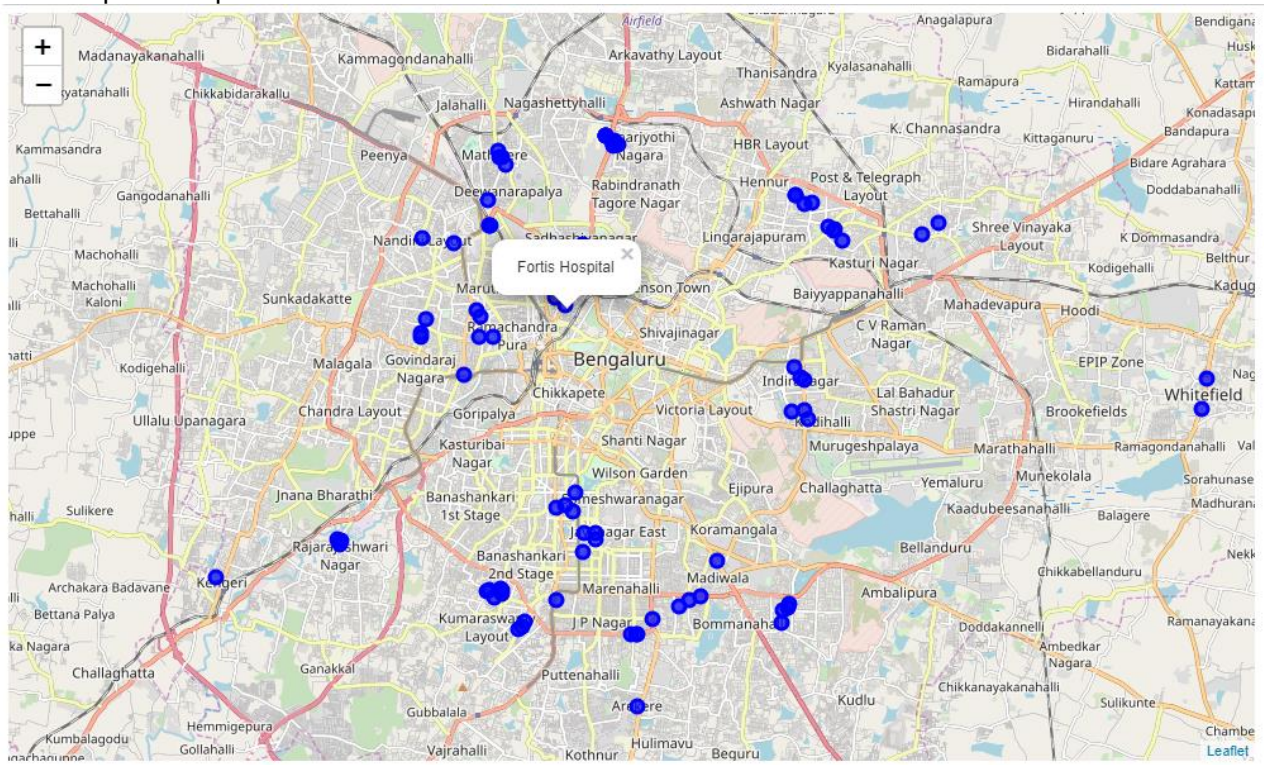
The food map:



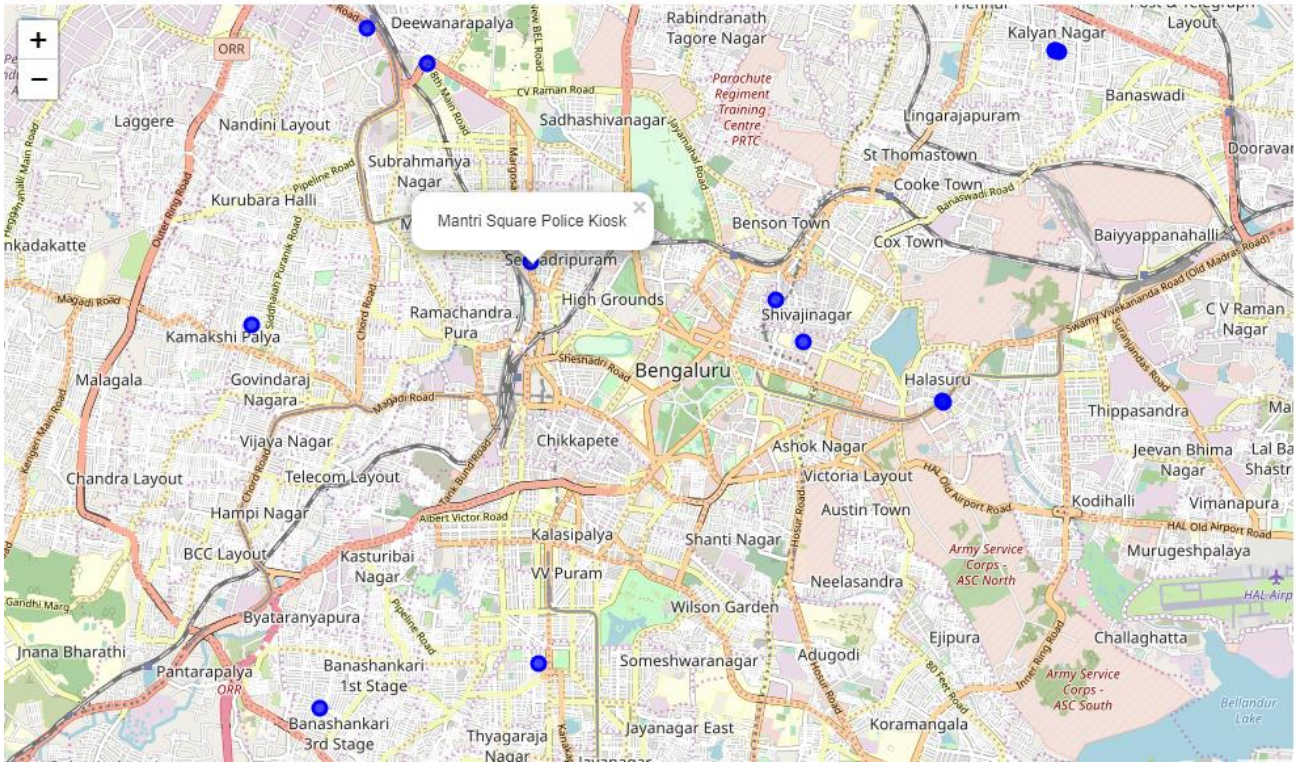
The Outdoor recreational Map:



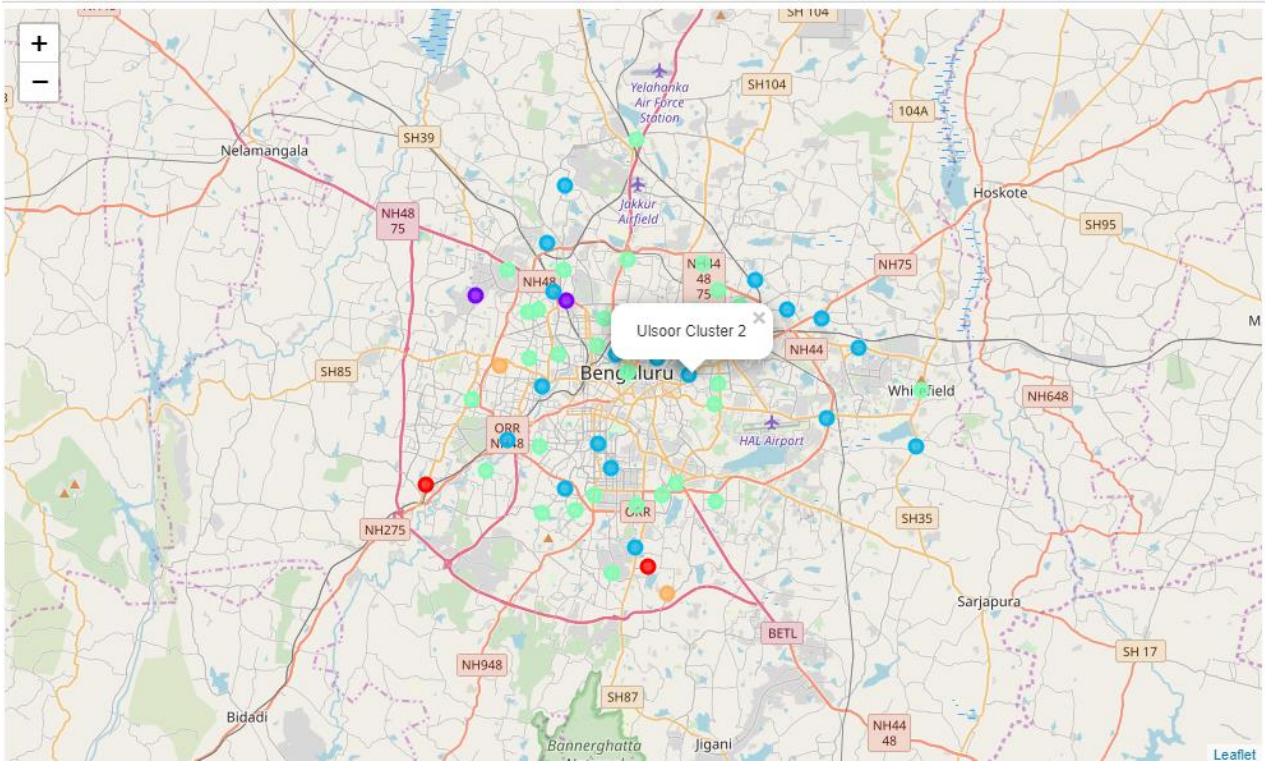
The Hospitals Map:



The Police Stations map:



The Cluster maps:



Discussions:

Now the visitors will be able to plan their visits according to various food locations in the city.
The last map in the result section is mainly for food eccentric visitors.
The clustering is based on top 10 restaurants of the location this can be changed for further clustering. The number of venues can be increased/decreased using the radius variable in API calls .
The cluster is helpful for people to identify similar places and not waste time in places similar kind f which they have already visited.

Conclusions:

This map is a great help to visitors (especially food eccentric visitors to the place). Bangalore is a city of people from several places and Backgrounds and it has something to offer to everyone.