Coursera Capstone Project: Applied Data Science

KSHITIJ BAJPAI

GCET, GREATER NOIDA

**INTRODUCTION**

Delhi is busy state and it is rapidly growing day by day. Until corona hits it has an increasing economy and jobs were literally flouring for a certain pay scale people. It is situated in northern area of India i.e. 28.7041° N, 77.1025° E.

Being busy and filled with approximately 2 cr. People in the area of 1484 km2 makes it crowded. There are certain malls and location for buying your stuff with quality items while there are also places where all items are cheap and dependable. Being one the most crowded area it suffers from traffic, busy line and crowded location which makes people who are in hurry to suffer a lot.

**BUSINESS PROBLEM**

In a busy location like Delhi it is hard to search a place which is near and has good rating while it should also be free to take in customers. This project will help those people who want have food and don’t wish to wait for long.

**DATA**

The data for this project is gathered from multiple locations. All the location are explained below:

**Places**

All the places or locations in Delhi are scraped from Wikipedia page ( <https://en.wikipedia.org/wiki/Neighbourhoods_of_Delhi> ). This give us all the places in delhi according to city.

source =

requests.get('https://en.wikipedia.org/wiki/Neighbourhoods\_of\_Delhi').text

soup = BeautifulSoup(source, 'lxml')

neighborhood=[]

for i in soup.find\_all("h2"):

for k in i.next:

flag=0

if "".join(list(k)) in ['See also','References','External links']:

flag=1

break

if flag==1:

continue

for j in i.next\_siblings:

if j.name=='ul':

text=j.text

delimiters = ",", ".", "[",";"

regexPattern = '|'.join(map(re.escape, delimiters))

for m in text.split('\n'):

neighborhood.append(re.split(regexPattern,m)[0])

break

**Coordinates**

To gather the coordinates using google maps there was a requirement for a billing account which I don’t have. So, all the coordinates of each location is scraped from google page using the crawler.

def coordinate\_extract(soup):

coor=[]

search = soup.find\_all(class\_='BNeawe s3v9rd AP7Wnd')

for i in search:

try:

c=float(i.text)

coor.append(c)

except:

continue

if len(coor)<2:

try:

for i in map(lambda x: re.findall(r'[0-9]{2}\.[0-9]{6}',x.text),search):

coor.extend(list(map(lambda x: float(x), i)))

if len(coor)<2:

raise exception('')

except:

for i in map(lambda x: re.findall(r'[0-9]{2}\.[0-9]{2,6}',x.text),search):

coor.extend(list(map(lambda x: float(x), i)))

finally:

coor=coor[:2]

return coor[0],coor[1]

# this is all done in accordance with the google page

**Venue**

The venue data is gathered using Foursquare API which is necessity of this project

explore\_df\_list = []

for i, nbd\_name in enumerate(df['Location']):

try :

# Getting the data of Neighbours

nbd\_name = df.loc[i, 'Location']

nbd\_lat = df.loc[i, 'Latitude']

nbd\_lng = df.loc[i, 'Longitude']

radius = 1000 # Setting the radius as 1000 metres

LIMIT = 50 # Getting the top 30 venues

url = 'https://api.foursquare.com/v2/venues/explore?client\_id={}&client\_secret={}&ll={},{}&v={}&radius={}&limit={}'\

.format(CLIENT\_ID, CLIENT\_SECRET, nbd\_lat, nbd\_lng, VERSION, radius, LIMIT)

results = json.loads(requests.get(url).text)

results = results['response']['groups'][0]['items']

nearby = json\_normalize(results) # Flattens JSON

# Filtering the columns

filtered\_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']

nearby = nearby.loc[:, filtered\_columns]

# Renaming the columns

columns = ['Name', 'Category', 'Latitude', 'Longitude']

nearby.columns = columns

# Gets the categories

nearby['Category'] = nearby.apply(get\_category\_type, axis=1)

**METHODOLGY**

# Gets the data required

for i, name in enumerate(nearby['Name']):

s\_list = nearby.loc[i, :].values.tolist() # Converts the numpy array to a python list

f\_list = [nbd\_name, nbd\_lat, nbd\_lng] + s\_list

explore\_df\_list.append(f\_list)

except Exception as e:

pass

Scraped the locations and grouped them according to their category type to get all the food venues.

meals=

['Restaurant','Place','Food','Joint','Breakfast','Hotel','Coffee','Donut','Dessert','Bakery','Diner','Café','Ice Cream','Cafeteria']

data=explore\_df[(explore\_df['Venue Category']).apply(lambda x: True if any(item in x.split() for item in meals) else False)==True ]

data.reset\_index(inplace=True,drop=True)

data.head(10)

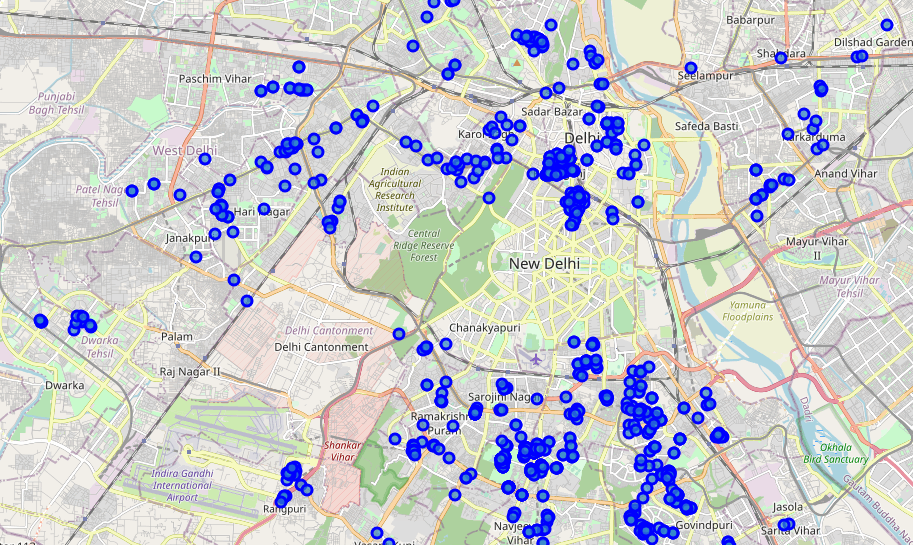
The objective is to find all the nearby places from where we can find the food locations

# Example coordinate

a1=28.6304

a2=77.2177

Get nearby locations of given example coordinates



from geopy.distance import distance

place={}

for i in zip(data['Venue Name'],data['Venue Latitude'],data['Venue Longitude']):

coor=(i[1],i[2])

k=distance(coor,(a1,a2))

if k<=2.000000000000000:

place[i[0]]=[k,i[1],i[2]]

places={k: v for k, v in sorted(place.items(), key=lambda item: item[1][0])}

**RESULTS**

Found nearest places

nearby\_venue\_latitude=[]

nearby\_venue\_longitude=[]

for i in places.items():

nearby\_venue\_latitude.append(i[1][1])

nearby\_venue\_longitude.append(i[1][2])

final=pd.DataFrame()

final['Name']=places.keys()

final['Latitude']=nearby\_venue\_latitude

final['Longitude']=nearby\_venue\_longitude

Mapping all nearby places

map\_nearby = folium.Map(location=[a1, a2], zoom\_start=14)

folium.Marker([a1,a2]).add\_to(map\_nearby)

for lat, lng, Location in zip(final['Latitude'], final['Longitude'], final['Name']):

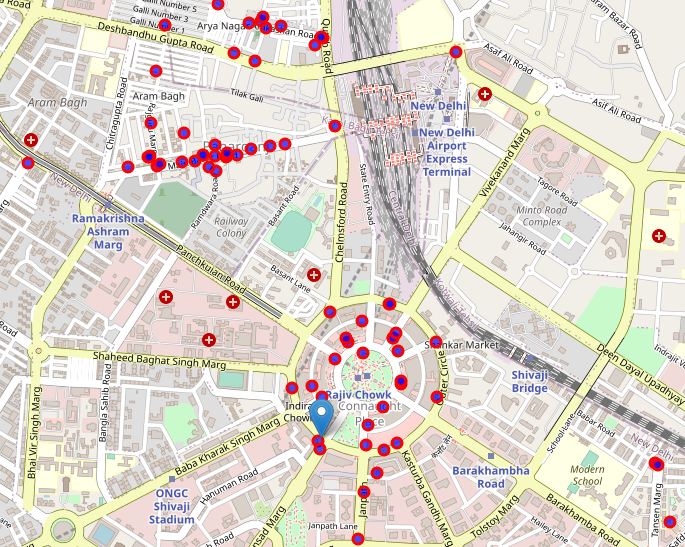
label = '{}'.format(Location)

label = folium.Popup(label, parse\_html=True)

folium.CircleMarker(

[lat, lng],radius=5, popup=label, color='red', fill=True, fill\_color='blue', fill\_opacity=0.7,

parse\_html=False).add\_to(map\_nearby)



**CONCLUSION**

If someone find the place they are in time consuming then they can move to next places that is nearest to them