Clustering Neighborhoods in Allahabad

Introduction to the Problem

We would try to implement the similar problem we have been taught and discussed in the course itself. We would try to find out that how similar or dissimilar two areas of a city are considering some specific features. For our case we are going to consider Allahabad, It was not easy to fing the allahabad dataset but still, we managed to collect it.

Solution

Here I will convert addresses to their corresponding latitude and longitude values. I will use the Foursquare API to explore neighborhoods in Allahabad. I will use the explore function to get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters. I will use the k-means clustering algorithm to complete this task. Finally, I will use the Folium library to visualize the neighborhoods in Allahabad City and their emerging clusters

Way to the Solution

- Download and Explore Dataset
- Explore Neighborhoods in Allahabad India
- Analyze Each Neighborhood
- Cluster Neighborhoods
- Examine Clusters

Installing all the required dependencies

In [1]:

```
!pip install geocoder
```

```
Requirement already satisfied: geocoder in /usr/loc
al/lib/python3.6/dist-packages (1.38.1)
Requirement already satisfied: future in /usr/loca
l/lib/python3.6/dist-packages (from geocoder) (0.1
6.0)
Requirement already satisfied: click in /usr/local/
lib/python3.6/dist-packages (from geocoder) (7.1.2)
Requirement already satisfied: ratelim in /usr/loca
l/lib/python3.6/dist-packages (from geocoder) (0.1.
6)
Requirement already satisfied: six in /usr/local/li
b/python3.6/dist-packages (from geocoder) (1.15.0)
Requirement already satisfied: requests in /usr/loc
al/lib/python3.6/dist-packages (from geocoder) (2.2
3.0)
Requirement already satisfied: decorator in /usr/lo
cal/lib/python3.6/dist-packages (from ratelim->geoc
oder) (4.4.2)
Requirement already satisfied: urllib3!=1.25.0,!=1.
25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dis
t-packages (from requests->geocoder) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /us
r/local/lib/python3.6/dist-packages (from requests-
>geocoder) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from reques
ts->geocoder) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 i
n /usr/local/lib/python3.6/dist-packages (from requ
ests->geocoder) (2020.6.20)
```

Import each and every required library and package

- BeautifulSoup and requests for scraping the data
- Pandas and numpy for making structure and preprocessing of the data
- Geopy for getting the long and lats of the places
- Folium for maps and more information
- Matplotlib for visualization
- Sklearn for KMeans model

In [2]:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
from geopy.geocoders import Nominatim

import numpy as np
import matplotlib.cm as cm
import matplotlib.colors as colors
import folium
from sklearn.cluster import KMeans
```

Scrapping of the datafrom the wikipedia page https://en.m.wikipedia.org/wiki/Neighbourhoods_of_Allaha (https://en.m.wikipedia.org/wiki/Neighbourhoods_of_Allaha

After doing the proper inspection of the page I got to know that the the names are stored under li tags inside a division of class 'mw-parser-output'.

In [3]:

```
data = requests.get("https://en.m.wikipedia.org/wiki/Neighbourhoods_of_Allahabad").text
soup = BeautifulSoup(data, 'html.parser')
neighborhoodList = []
for row in soup.find_all("div", class_= "mw-parser-output")[0].findAll("ul")[0].findAll("li"):
  neighborhoodList.append(row.text.split()[1])
kl_df = pd.DataFrame({"Neighborhood": neighborhoodList})
kl_df.head()
print(len(kl_df))
```

```
53
```

Geolocation coordinates generation of the places

In [4]:

```
geolocator = Nominatim(user_agent="courcera_capston")
new_list = []
def get_latlng(neighborhood):
    global new_list
    location = geolocator.geocode('{}, Prayagraj, India'.format(neighborhood))
    try:
      loc = (location.latitude, location.longitude)
      new_list.append(neighborhoodList)
      return loc
    except:
      pass
coords = [get_latlng(neighborhood) for neighborhood in kl_df["Neighborhood"].tolist() if get_latlng(neighborhood) != None]
```

Get the location of the city Allahabad and combning them to the location data frame.

In [5]:

```
address = 'Prayagraj, India'
geolocator = Nominatim(user_agent="my-application")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Hyderabad, India {}, {}.'.f
ormat(latitude, longitude))
```

```
The geograpical coordinate of Hyderabad, India 25.4
381302, 81.8338005.
```

In [6]:

```
df_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitud
e'])
kl_df['Latitude'] = df_coords['Latitude']
kl_df['Longitude'] = df_coords['Longitude']
kl_df.dropna(inplace=True)
print(kl_df.shape)
```

```
(22, 3)
```

Plot the datapoints of the dataframe on the map using folium

In [7]:

```
map_kl = folium.Map(location=[latitude, longitude], zoom_start=11)
for lat, lng, neighborhood in zip(kl_df['Latitude'],  kl_df['Longitude'], kl_df['Neighborhood']):
 label = '{}'.format(neighborhood)
 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).add_to(map_kl)
map_kl
```

Out[7]:

Connecting to the foursquare api to get more info about the locations

In [8]:

```
CLIENT_ID = 'JH54IDPYRYILFWBGNXRIB2UXSNYGDGUJVHKPROH44R0TLGII'
CLIENT_SECRET = '1C0YP3ZVJP3ZS3VOQEWAUP4DJM5TBBBHMTIFUTCEAGYZQKBM'
VERSION = '20180605'
radius = 2000
LIMIT = 100
venues = []
for lat, long, neighborhood in zip(kl_df['Latitude'], kl_df['Longitude'], kl_df['Neighborhood']):
  url = "https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}".format(CLIENT_ID,CLIENT_SECRET,VERSION,lat,long,radius,LIMIT)
  results = requests.get(url).json()["response"]['groups'][0]['items']
  for venue in results:
      venues.append((neighborhood,lat,long,venue['venue']['name'],
      venue['venue']['location']['lat'],venue['venue']['location']['lng'],venue['venue']['categories'][0]['name']))
```

In [9]:

```
venues_df = pd.DataFrame(venues)
venues_df.columns = ['Neighborhood', 'Latitude', 'Longitude', 'V
enueName', 'VenueLatitude', 'VenueLongitude', 'VenueCategory']
print(venues_df.shape)
venues_df.head()
```

```
(157, 7)
```

Out[9]:

	Neighborhood	Latitude	Longitude	VenueName	VenueLatit
0	Allengunj	25.470439	81.821637	Prayag Inn	25.460
1	Allengunj	25.470439	81.821637	The Legend Hotel	25.459
2	Alopibagh	25.431367	81.830679	Chowk	25.440
3	Alopibagh	25.431367	81.830679	cafe coffee day	25.445
4	Alopibagh	25.431367	81.830679	Woodland Original	25.433

In [10]:

```
print('There are {} unique categories.'.format(len(venues_df['VenueCategory'].unique())))
venues_df['VenueCategory'].unique()
```

```
There are 23 unique categories.
```

Out[10]:

```
array(['Indian Restaurant', 'Hotel', 'Flea Market',
'Coffee Shop',
       'Clothing Store', 'Garden', 'ATM', 'Train St
ation',
       'Historic Site', 'River', 'Restaurant', 'Mul
tiplex',
       'Fast Food Restaurant', 'Shopping Mall', 'Ca
fé', 'Pizza Place',
       'Pharmacy', 'Business Service', 'Tennis Cour
t', 'Moving Target',
       'Department Store', 'Indie Movie Theater',
'Airport'], dtype=object)
```

In [11]:

```
# One hot encoding of the l
kl_onehot = pd.get_dummies(venues_df[['VenueCategory']], prefix=
"", prefix_sep="")
# Adding neighborhood column back to dataframe
kl_onehot['Neighborhoods'] = venues_df['Neighborhood']
# Moving neighbourhood column to the first column
fixed_columns = [kl_onehot.columns[-1]] + list(kl_onehot.columns
[:-1])
kl_onehot = kl_onehot[fixed_columns]
print(kl_onehot.head())
```

```
  Neighborhoods  ATM  Airport  ...  Shopping Mall
Tennis Court  Train Station
0     Allengunj    0        0  ...              0
0              0
1     Allengunj    0        0  ...              0
0              0
2     Alopibagh    0        0  ...              0
0              0
3     Alopibagh    0        0  ...              0
0              0
4     Alopibagh    0        0  ...              0
0              0

[5 rows x 24 columns]
```

In [12]:

```
kl_grouped=kl_onehot.groupby(["Neighborhoods"]).sum().reset_index()
print(kl_grouped.shape)
kl_grouped.head()
```

```
(22, 24)
```

Out[12]:

	Neighborhoods	ATM	Airport	Business Service	Café	Clothing Store	Co S
0	Allengunj	0	0	0	0	0	
1	Alopibagh	0	0	0	0	1	
2	Ashok	2	0	0	0	0	
3	Atala	0	0	0	0	0	
4	Attarsuiya	0	0	0	0	0	

In [13]:

```
# Creating a dataframe for Shopping Mall data only
kl_mall = kl_grouped[["Neighborhoods","Shopping Mall"]]
```

In [16]:

```
kclusters = 2
kl_clustering = kl_mall.drop(["Neighborhoods"], 1)
# Run k-means clustering algorithm



kmeans = KMeans(n_clusters=kclusters,random_state=0).fit(kl_clus
tering)
# Checking cluster labels generated for each row in the datafram
e


kmeans.labels_[0:10]
```

Out[16]:

```
array([0, 0, 0, 0, 0, 1, 1, 0, 1, 0], dtype=int32)
```

In [17]:

```
# Creating a new dataframe that includes the cluster as well as
 the top 10 venues for each neighborhood.
kl_merged = kl_mall.copy()


# Add the clustering labels
kl_merged["Cluster Labels"] = kmeans.labels_
kl_merged.rename(columns={"Neighborhoods": "Neighborhood"}, inpl
ace=True)
kl_merged.head(10)
```

Out[17]:

	Neighborhood	Shopping Mall	Cluster Labels
0	Allengunj	0	0
1	Alopibagh	0	0
2	Ashok	0	0
3	Atala	0	0
4	Attarsuiya	0	0
5	Bahadurgunj	1	1
6	Bai-Ka-Bagh	1	1
7	Bairahana	0	0
8	Beli	1	1
9	Benigunj	0	0

In [18]:

```
# Adding latitude and longitude values to the existing dataframe
kl_merged['Latitude'] = kl_df['Latitude']
kl_merged['Longitude'] = kl_df['Longitude']
# Sorting the results by Cluster Labels
kl_merged.sort_values(["Cluster Labels"], inplace=True)
kl_merged
```

Out[18]:

	Neighborhood	Shopping Mall	Cluster Labels	Latitude	Longitude
0	Allengunj	0	0	25.470439	81.821637
19	Georgetown	0	0	25.435767	81.891112
17	Dariyabad	0	0	25.439489	81.849859
16	Darbhanga	0	0	25.428468	81.811993
12	Chowk	0	0	25.477762	81.846038
11	Chatham	0	0	25.465610	81.852011
9	Benigunj	0	0	25.483881	81.875911
10	Bharadwaj	0	0	25.427540	81.815529
4	Attarsuiya	0	0	25.469797	81.862827
3	Atala	0	0	25.457509	81.858548
2	Ashok	0	0	25.438182	81.857905
1	Alopibagh	0	0	25.431367	81.830679
7	Bairahana	0	0	25.445406	81.883378
21	Johnstongunj	0	0	25.448991	81.734161
8	Beli	1	1	25.453753	81.854000
6	Bai-Ka-Bagh	1	1	25.454475	81.833826
20	Govindpur	1	1	25.438130	81.833800
5	Bahadurgunj	1	1	25.437593	81.835918
13	Civil	1	1	25.438130	81.833800
14	Colonelgunj	1	1	25.438130	81.833800
15	Daraganj	1	1	25.438130	81.833800
18	Dhoomangunj	1	1	25.446209	81.844576

In [19]:

```
# Creating the map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)
# Setting 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(kl_merged['Latitude'], kl_merged['Longitude'], kl_merged['Neighborhood'], kl_merged['Cluster Labels']):
  label = folium.Popup(str(poi) + ' - Cluster ' + str(cluster), parse_html=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
```

Out[19]:

In [20]:

```
print(len(kl_merged.loc[kl_merged['Cluster Labels'] == 0]))
print(len(kl_merged.loc[kl_merged['Cluster Labels'] == 1]))
```

```
14
8
```

In [ ]:

```

```