Capstone Project - The Battle of the Neighbourhoods

Satyajeet A. Dhawale
satyajit.dhawale@gmail.com
September 6th, 2020

1. **Introduction**

Nagpur is central India biggest city and has the number of interesting places within the boundaries of the city. There are many places, restaurant and malls in Nagpur, where many people visit on daily basis. Also, due to new style of food delivery many people here have opened several home based small food restaurants and people prefer ordering from them. Hence to succeed with retail there has to better way of selecting a nearby restaurant and provide a fast accessible experience.

2. **Business Problem**

Now and then everyone wants to eat from outside and also people go for business lunch/dinner or required food for partying at home/office. The main idea is to find the ideal nearby, optimal and most density restaurant for the customer base in city.

This will help both customer and the business owner. For customers they will easily know what restaurant or food place is nearby them. And for business owner they can calculate they will know in which area that particular type of restaurant I not present.

3. **Data**

All the major locations in Nagpur city were taken from the Wikipedia page (https://en.wikipedia.org/wiki/List_of_localities_in_Nagpur) and scraped using BeautifulSoup library in Python. To get the latitude and longitude of each location I have used Geocoder library in python and stored in csv file for each location.

The venue data is then found via the FourSquare API by passing coordinates of each location. And all the venue data is captured in another new DataFrame.

Code – Scrapping Neighbourhoods data from Wikipedia using Beautiful Soup

```
#! /usr/bin/python

source =
requests.get('https://en.wikipedia.org/wiki/List_of_localities_in_Nagpur').text
soup = BeautifulSoup(source, 'lxml')

csv_file = open('Nagpur.csv', 'w')
csv_writer = csv.writer(csv_file)
csv_writer.writerow(['Neighbourhood'])

list_items = soup.find_all('li')
list_items

for i in list_items[0:42]:
    temp = i.text.replace('\n','')
    print(temp)
    csv_writer.writerow([temp])
csv_file.close()
```

Some of the Neighbourhoods have named incorrectly (with description and reference link) so I have to rename them. As shown in figure and code snippet below.

	Neighbourhood
0	Mahal — The oldest locality in Nagpur. Nagpur ...
1	Sitabuldi[2]
2	Dhantoli[3]
3	Itwari[4]
4	Mominpura[5]

Figure-1: Neighborhoods with Incorrect Names

Code – Renaming the incorrect Neighbourhood Names

```
#! /usr/bin/python

df.loc[df['Neighbourhood']== 'Mahal — The oldest locality in Nagpur. Nagpur was
founded here by Raja Bakht Buland Shah. The Bhonsle Rajwada is also located
here.'] = 'Mahal'
df.loc[df['Neighbourhood']== 'Sitabuldi[2]'] = 'Sitabuldi'
df.loc[df['Neighbourhood']== 'Dhantoli[3]'] = 'Dhantoli'
df.loc[df['Neighbourhood']== 'Itwari[4]'] = 'Itwari'
df.loc[df['Neighbourhood']== 'Mominpura[5]'] = 'Mominpura'
df.loc[df['Neighbourhood']== 'Gaddi Godam'] = 'Gaddigodam'
```

Code – Extracting the Latitude and Longitude for each Neighbourhood

```
#! /usr/bin/python

# Extracting Lat Long from Geocoder for each Neighbourhood

latitudes = [] # Initializing the latitude list
longitudes = [] # Initializing the longitude list

for location in df["Neighbourhood"] :
    place_name = location+', Nagpur, India'# Formats the place name
    print(place_name)

    time.sleep(250)
    g = geocoder.arcgis(place_name)
    lat_lng_coords = g.latlng
    print(lat_lng_coords)

    lat = lat_lng_coords[0] # Extracts the latitude value
    lng = lat_lng_coords[1] # Extracts the longitude value

    latitudes.append(lat) # Appending to the list of latitudes
    longitudes.append(lng) # Appending to the list of longitudes

df['Latitude'] = latitudes
df['Longitude'] = longitudes
```

Code – Finding the Venue data from FourSquare API

```
#! /usr/bin/python

nearby_df = pd.DataFrame()

for i, nbd_name in enumerate(df['Neighbourhood']):
        print(nbd_name)

        nbd_name = df.loc[i, 'Neighbourhood']
        nbd_lat = df.loc[i, 'Latitude']
        nbd_lng = df.loc[i, 'Longitude']

        radius = 1000 # Setting the radius as 1000 metres
        LIMIT = 30 # 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 = pd.json_normalize(results) # Flattens JSON
        nearby.rename(columns = {'venue.name':'Venue_Name',
'venue.location.lat':'Venue_Latitude',
                        'venue.location.lng':'Venue_Longitude',
'venue.categories':'Category'}, inplace = True)


        nearby['Neighbourhood'] = nbd_name
        nearby['Neighbourhood_Latitude'] = nbd_lat
        nearby['Neighbourhood_Longitude'] = nbd_lng
        nearby_df = nearby_df.append(nearby)

nearby_df.reset_index(drop=True, inplace=True)
nearby_df.head(30)
```

	Neighbourhood	Neighbourhood_Latitude	Neighbourhood_Longitude	Venue_Name	Venue_Category	Venue_Latitude	Venue_Longitude
0	Mahal	21.14578	79.10713	Jagdish sauji	Indian Restaurant	21.144353	79.098415
1	Mahal	21.14578	79.10713	raam bhandar	Breakfast Spot	21.147359	79.108110
2	Mahal	21.14578	79.10713	India Sun	Hotel	21.149407	79.108314
3	Mahal	21.14578	79.10713	Gandhi Bagh Park	Garden	21.151502	79.105705
4	Mahal	21.14578	79.10713	Karan Kothari Jewellers	Jewelry Store	21.153513	79.110741

Figure-2: nearby_df data frame showing Neighborhoods, their Latitude, Longitude, Vanue Name and it's Category extracted from ForeSquare API with Latitude, Longitude extracted from Geocoder.

4. Methodology

Precision of the Geocoder library in Python: It was noted that for some location the Geocoder library was giving the incorrect coordinates. And hence few of the locations has to be checked manually and changes because they were having same name as other locations in India. Also, some names were incorrectly mention in the Wikipedia that also needs to be corrected.

Folium: Folium is used to display the location points on the virtual map and for the cluster visualization.

Code – Folium code to Display the Nagpur map and its Neighbourhoods

```
#! /usr/bin/python

# Nagpur latitude and longitude using Google search
nagpur_lat = 21.1458
nagpur_lng = 79.0882
# Creates map of Nagpur using latitude and longitude values
nagpur_map = folium.Map(location=[nagpur_lat, nagpur_lng],
zoom_start=12)
# Add markers to map
for lat, lng, neighbourhood in zip(df['Latitude'], df['Longitude'],
df['Neighbourhood']):
    label = '{}'.format(neighbourhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.5,
        parse_html=False).add_to(nagpur_map)
nagpur_map
```

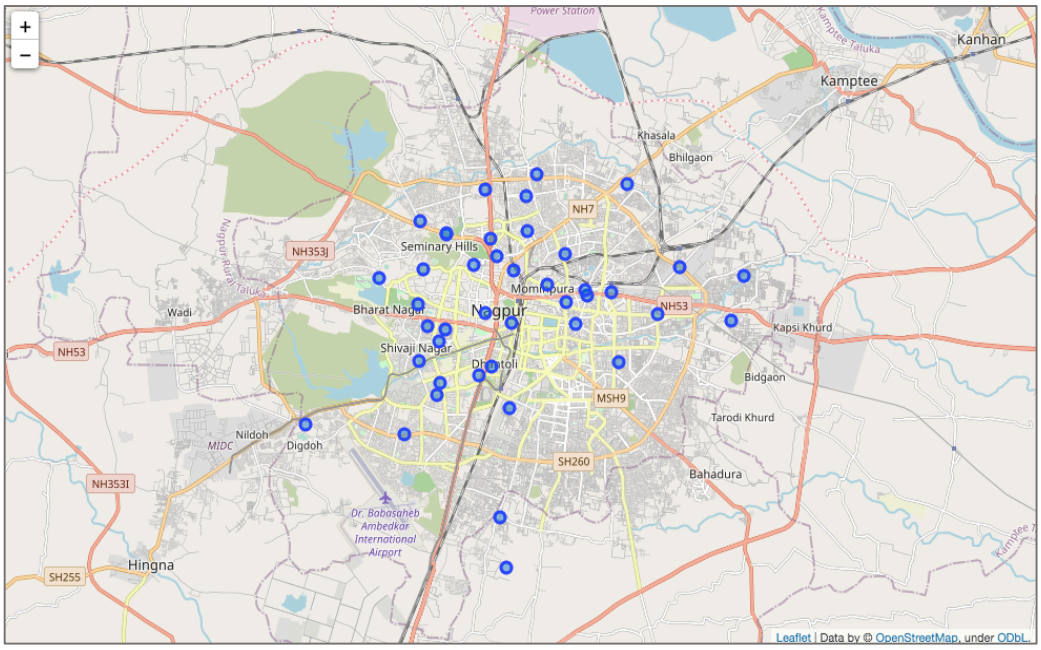

Figure-3: Nagpur map and its Neighborhoods plotted using Folium

One hot encoding: One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. For the K-means Clustering Algorithm, all unique items under Venue Category are one-hot encoded.

Code – One Hot Encoding Code

```
#! /usr/bin/python

# One hot encoding
nagpur_onehot = pd.get_dummies(explore_df[['Venue_Category']],
prefix="", prefix_sep="")

# Add neighborhood column back to dataframe
nagpur_onehot['Neighbourhood'] = explore_df['Neighbourhood']

# Move neighborhood column to the first column
fixed_columns = [nagpur_onehot.columns[-1]] + nagpur_onehot.columns[:-
1].values.tolist()
nagpur_onehot = nagpur_onehot[fixed_columns]

nagpur_onehot.head()
```

	Neighbourhood	ATM	Asian Restaurant	Athletics & Sports	BBQ Joint	Bakery	Bar	Bookstore	Breakfast Spot	Burger Joint	Business Service	Café	Campground	Chinese Restaurant	Clothing Store	Coffee Shop
0	Mahal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Mahal	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	Mahal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Mahal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Mahal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure-4: One Hot encoded DataFrame

Top 10 most common venues: Due to high variety in the venues, only the top 10 common venues are selected and a new Data Frame is made, which is used to train the K-means Clustering Algorithm.

Code – Top 10 most Common Venues

```
#! /usr/bin/python

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]

num_top_venues = 10
indicators = ['st', 'nd', 'rd']

# Create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1,
indicators[ind]))
    except:
```

```
        columns.append('{}th Most Common Venue'.format(ind+1))

# Create a new dataframe
neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted['Neighbourhood'] =
nagpur_grouped['Neighbourhood']

for ind in np.arange(nagpur_grouped.shape[0]):
    neighbourhoods_venues_sorted.iloc[ind, 1:] =
return_most_common_venues(nagpur_grouped.iloc[ind, :], num_top_venues)

neighbourhoods_venues_sorted.head()
```

Optimal number of clusters: Silhouette Score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Based on the Silhouette Score of various clusters below 20, the optimal cluster size is determined.

Code – Optimal Numbers of Clusters

```
#! /usr/bin/python

from sklearn.metrics import silhouette_samples, silhouette_score

indices = []
scores = []

for ngp_clusters in range(2, max_range) :

    # Run k-means clustering
    ngc = nagpur_grouped_clustering
    kmeans = KMeans(n_clusters = ngp_clusters, init = 'k-means++',
random_state = 0).fit_predict(ngc)

    # Gets the score for the clustering operation performed
    score = silhouette_score(ngc, kmeans)

    # Appending the index and score to the respective lists
    indices.append(ngp_clusters)
    scores.append(score)
```

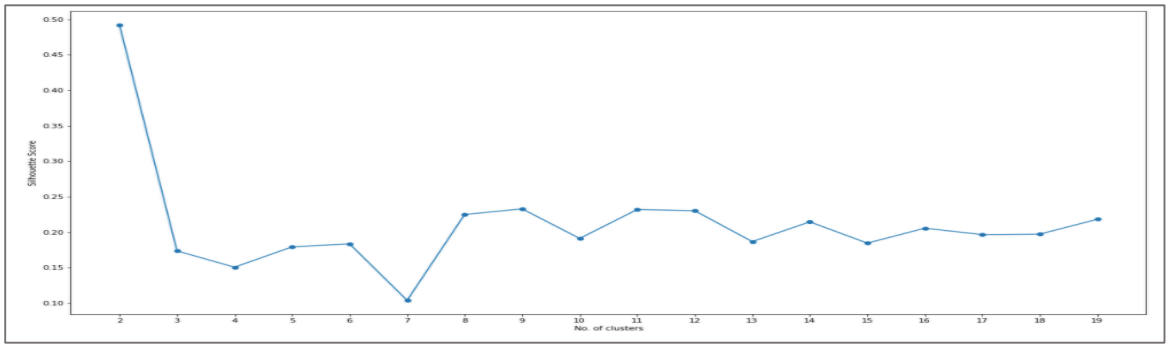

Figure-5: Silhouette Score vs Number of Clusters

K-means clustering: The venue data is then trained using K-means Clustering Algorithm to get the desired clusters to base the analysis on. K-means was chosen as the variables (Venue Categories) are huge, and in such situations K-means will be computationally faster than other clustering algorithms.

Code – Optimal Numbers of Clusters

```
#! /usr/bin/python

# Run k-means clustering
ngc = nagpur_grouped_clustering
kmeans = KMeans(n_clusters = ngp_clusters, init = 'k-means++',
random_state = 0).fit(ngc)
```

5. Results

The neighbourhoods are divided into ‘n’ clusters where ‘n’ is the number of clusters found using the optimal approach. The clustered neighbourhoods are visualized using different colors so as to make them distinguishable.

The six places i.e. Dharampeth, Ravi Nagar, Pratap Nagar, Gokulpeth, Giripeth and Gandhinagar area fall in very dense areas of Nagpur and are mostly surrounded with restaurants, ice cream shops, clothing stores, coffee and snacks places.

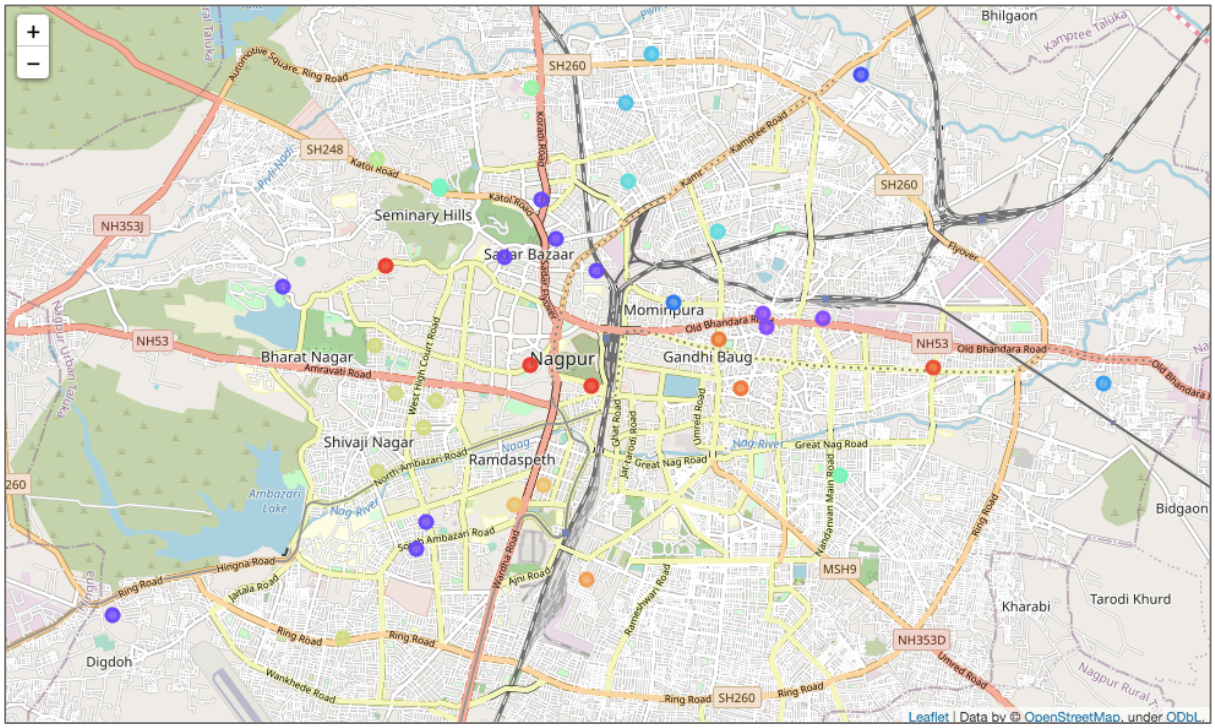

Figure-6: Clustred Neighbourhoods of Nagpur

6. Discussion

After analysing the various clusters produced by the Machine learning algorithm, cluster no.14, is a prime fit to that shows six different locations having different food corners that includes restaurants, ice cream shops, clothing stores, coffee and snacks places.

These areas fall in very dense areas of Nagpur and are mostly surrounded with people, hence it is obvious that most of the people will go here only. But there is a lot of scope for food business owners to open their food corners in areas other than these six areas.

7. Conclusion

As the high growing population of the Nagpur City and keeping in mind that India is a country of youngster's food corners like restaurants, ice cream shops, coffee and snacks places and clothing stores will have high demand.

If the business owners try to focus on the areas where there is a high demand of customers and less food corners their business will definitely will grow. Also, the it will be very beneficial for the people if the food corners will be near to their home.