**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.

1. **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.

1. **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.

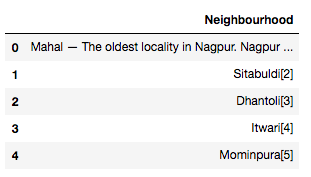


Figure-1: Neighborhoods with Incorrect Names

|  |
| --- |
| **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) |

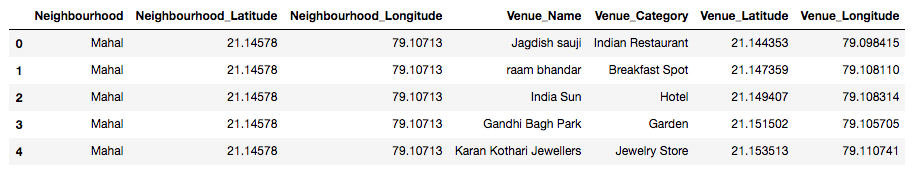
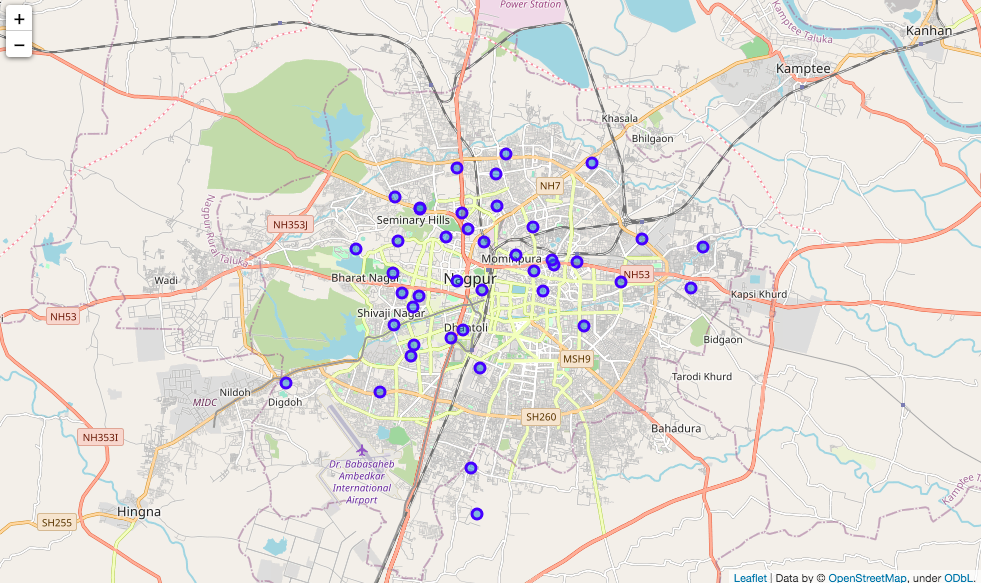


Figure-2: nearby\_df data frame showing Neighborhoods, their Latitude, Longitude, Vanue Name and it’s Category extracted from ForeSquare API with Latitude, Longitude extracted from Geocoder.

1. **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() |

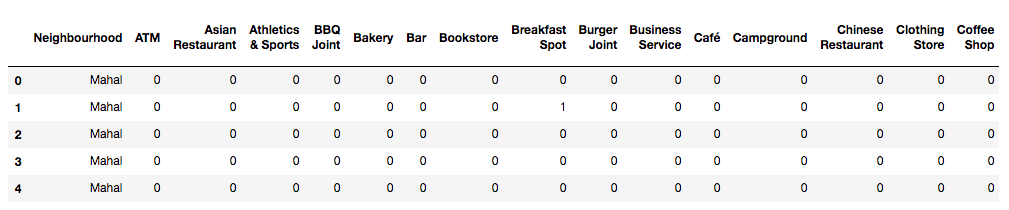


Figure-4: One Hot encoded DataFrame

**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) |

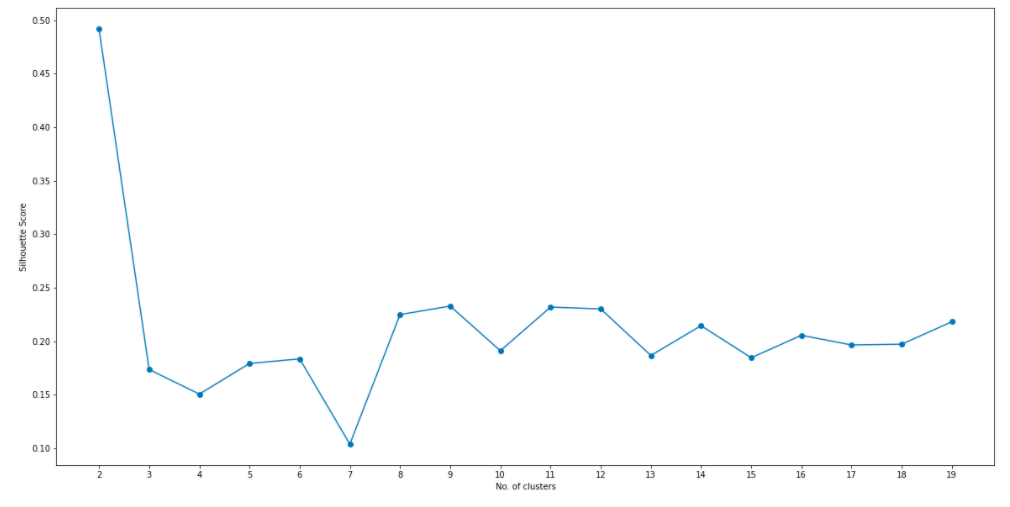


Figure-5: Silhouette Score vs Number of Clusters

**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) |

1. **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.

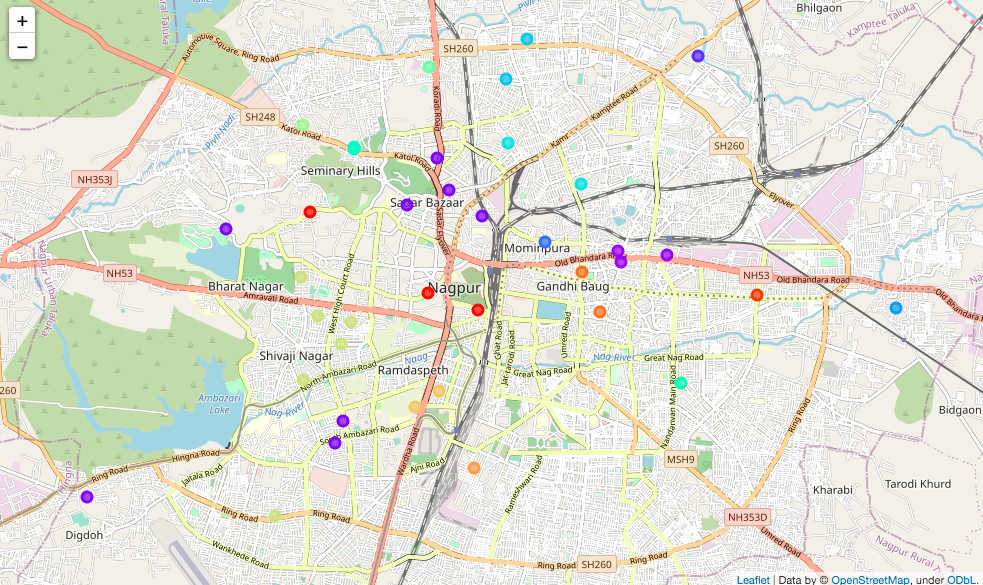
****

Figure-6: Clustred Neighbourhoods of Nagpur

1. **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.

1. **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.