Top 10 Places to Visit near you: Capstone project

Abhishek Sinha
IT Engineer, UK

1 Introduction

Kolkata, also known as Calcutta, is the capital of the Indian state of West Bengal. According to the 2011 Indian census, it is the seventh most populous city in India; the city had a population of 4.5 million, while the suburb population brought the total to 14.1 million, making it the third-most populous metropolitan area in India.

Kolkata district, which occupies an area of 185 km2 (71 sq mi), had a population of 4,486,679; its population density is 24,252/km2 (62,810/sq mi).

Kolkata is densely populated city spreading over a vast area with cultural diversity. Kolkata has been called the "City of Furious, Creative Energy" as well as the "cultural capital of India".

Kolkata see a heavy influx of tourist throughout the year due to its diversity in Food, culture, language, art and heritage buildings and museums.

The objective of this project is to provide best venues or location to visit, along with important places like banks., ATMs, Hospitals etc. for tourists each Neighbourhood using Data Science Methodology and Machine Learning algorithm.

2 Business Problem

Kolkata is a big city spanning over several kilometres in radius. Being one of the oldest city Kolkata has a complex system of roads and intersections.
Often tourists find themselves lost now and then if one wrong turn is made while travelling within in the city. It's very difficult to roam around the city without a guide or without taking help from local people asking about where what lies.

This project will help in generating Top 10 Venues/places to visit/important places like banks, hospitals, ATM etc. in each neighbourhood, which can be used as a system of providing information for tourist on what lies in their neighbourhood.

3 Data

The data for this project has been retrieved and processed through multiple sources, giving careful considerations to the accuracy of the methods used.

3.1 Neighbourhoods

The data of the neighbourhoods in Kolkata can be extracted out by web scraping using *BeautifulSoup* library for Python. The neighbourhood data is scraped from a *Wikipedia* page.

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

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

mwcg = soup.find_all(class_ = "mw-category-group")

length = len(mwcg) # Gets the length of number of `mw-category-groups` present

for i in range(1, length):
   lists = mwcg [i].find_all('a')
   for list in lists:
      nbd = list.get('title')
      csv_writer.writerow([nbd])

csv_file.close()
```

3.2 Geocoding

The file contents from **Kolkata.csv** is retrieved into a ***Pandas*** Dataframe. The latitude and longitude of the neighbourhoods are retrieved using **Google Maps Geocoding API**. The geometric location values are then stored into the initial dataframe.

```
latitudes = [] # Initializing the latitude array
longitudes = [] # Initializing the longitude array

for nbd in df["Neighbourhood"] :
   place_name = nbd + ",Kolkata,India" # Formats the place name
   url =
'https://maps.googleapis.com/maps/api/geocode/json?address={}&key={}'.format(place_na
me, API_KEY) # Gets the proper url to make the API call
   obj = json.loads(requests.get(url).text) # Loads the JSON file in the form of a python
dictionary

   results = obj['results'] # Extracts the results information out of the JSON file
   lat = results[0]['geometry']['location']['lat'] # Extracts the latitude value
   lng = results[0]['geometry']['location']['lng'] # 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
```

3.3 Venue Data

From the location data obtained after Web Scraping and Geocoding, the venue data is found out by passing in the required parameters to the **FourSquare API** and creating another Dataframe to contain all the venue details along with the respective neighbourhoods.

```
explore_df_list = []
for i, nbd_name in enumerate(df['Neighbourhood']):

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

      radius = 1000
      LIMIT = 30

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)

filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',
'venue.location.lng']
nearby = nearby.loc[:, filtered_columns]

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

nearby['Category'] = nearby.apply(get_category_type, axis=1)

for i, name in enumerate(nearby['Name']):
s_list = nearby.loc[i, :].values.tolist() f_list = [nbd_name, nbd_lat, nbd_lng] + s_list
explore_df_list.append(f_list)

   except Exception as e:
      pass
```

4 Methodology

A thorough analysis of the principles of methods, rules, and postulates employed g=have been made in order to ensure the inferences to be made are as accurate as possible.

4.1 Accuracy of the Geocoding API

In the initial development phase with OpenCage Geocoder API, the number of erroneous results were of an appreciable amount, which led to the development of an algorithm to analyse the accuracy of the Geocoding API used. In the algorithm developed, Geocoding API from various providers were tested, and in the end, Google Maps Geocoder API turned out to have the least number of collisions (errors) in our analysis.

```
col = 0
explored_lat_lng = []
for lat, lng, neighbourhood in zip(df['Latitude'], df['Longitude'], df['Neighbourhood']):

if (lat, lng) in explored_lat_lng: col = col + 1

else:
explored_lat_lng.append((lat, lng))

print("Collisions : ", col)
```

4.2 Folium

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the **leaflet.js** library. All cluster visualizations are done with help of Folium which in turn generates a Leaflet map made using *OpenStreetMap* technology.

```
kol_lat = 22.5726
kol_lng = 88.3639

map_kolkata = folium.Map(location=[kol_lat, kol_lng], zoom_start=10)

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=0.7,
        parse_html=False).add_to(map_kolkata)
```

Out[18]:

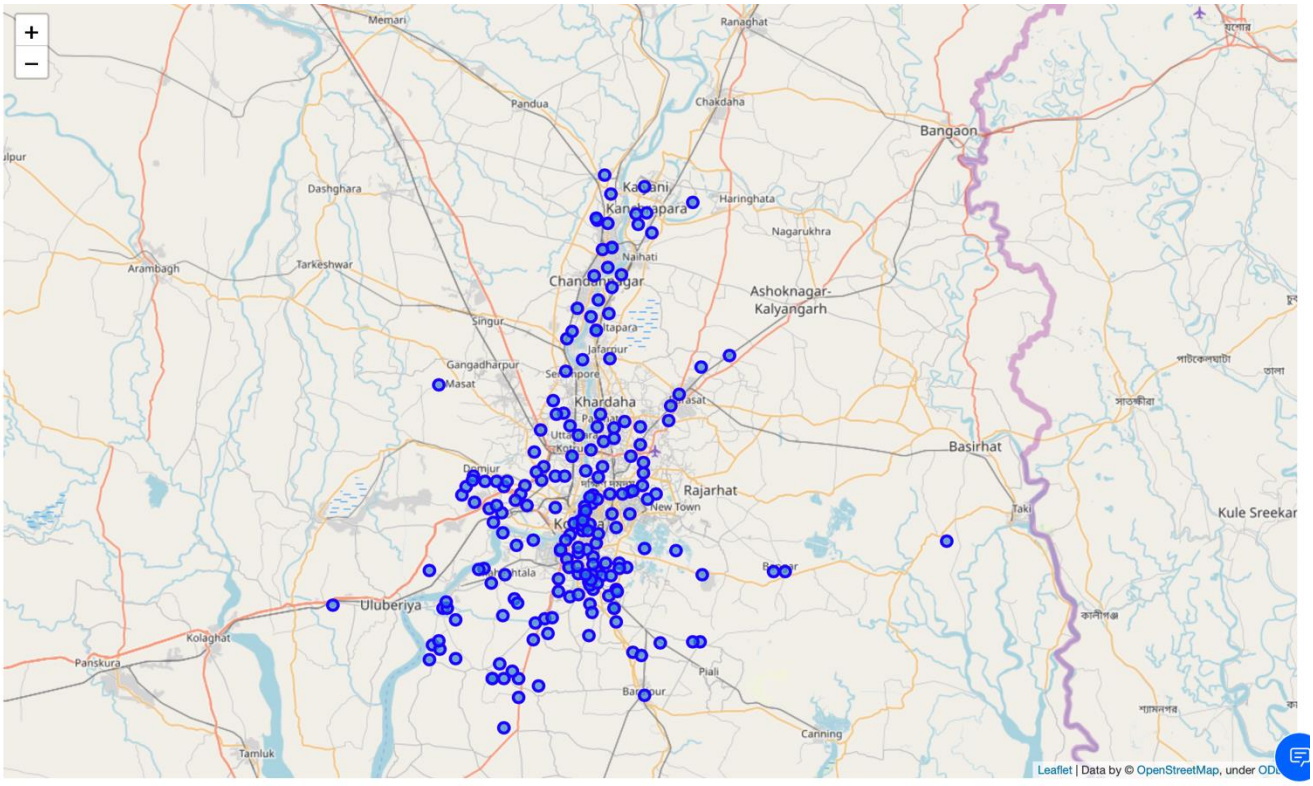


Figure 1 – Kolkata Neighbourhoods

4.3 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.

```
kolkata_onehot = pd.get_dummies(explore_df[[‘Venue Category’]],
prefix="", prefix_sep="")
kolkata_onehot[‘Neighbourhood’] = explore_df[‘Neighbourhood’]

fixed_columns = [kolkata_onehot.columns[-1]] + kolkata_onehot.columns[:-1]. values.tolist()
kolkata_onehot = kolkata_onehot[fixed_columns]

kolkata_grouped = kolkata_onehot.groupby(‘Neighbourhood’).mean().reset_index()
```

4.4 Top 10 most common venues

Due to high variety in the venues, only the top 10 common venues are selected and a new DataFrame is made, which is used to train the K-means Clustering Algorithm

```
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']

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))

neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted['Neighbourhood'] = kolkata_grouped['Neighbourhood']

for ind in np.arange(kolkata_grouped.shape[0]):

 neighbourhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues
(kolkata_grouped.iloc[ind, :], num_top_venues)
```

4.5 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 neighbouring clusters. Based on the Silhouette Score of various clusters below 20, the optimal cluster size is determined.

```
import matplotlib.pyplot as plt
%matplotlib inline
def plot(x, y, xlabel, ylabel):
        plt.plot(x, y, 'o-')
        plt.figure(figsize = (20,10))
        plt.xlabel("No. of clusters")
        plt.ylabel("Silhouette Score")
        plt.show()
```

```
In [30]: plot(max_range, scores, "No. of clusters", "Silhouette Score")
```

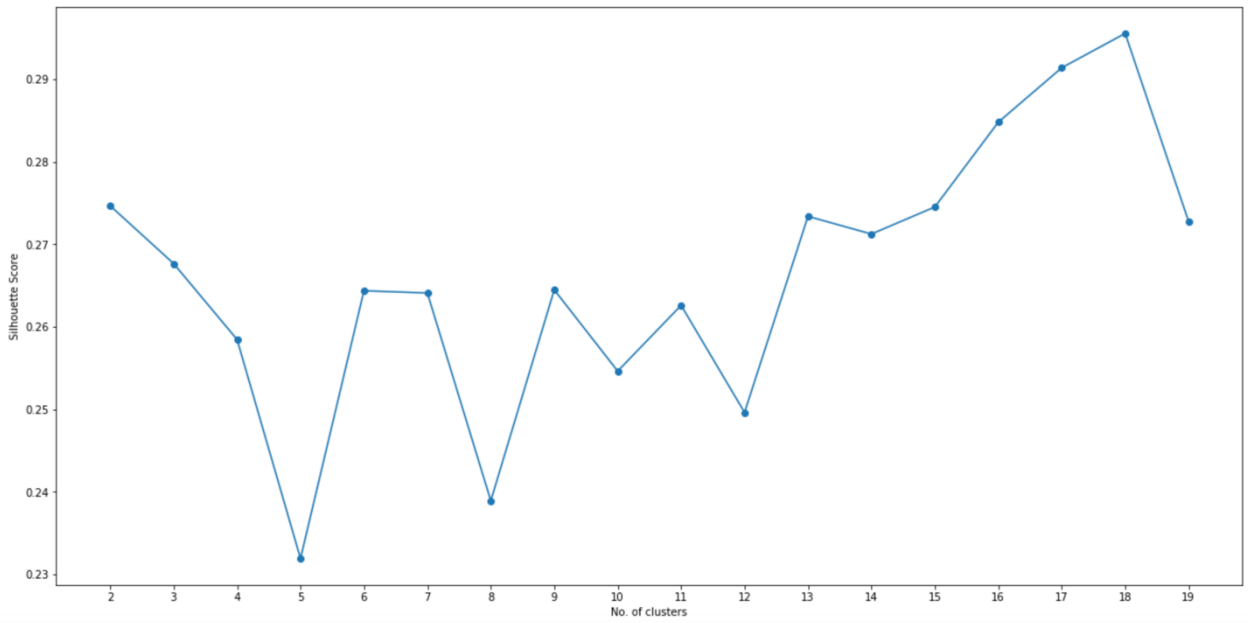


Figure 2 – Silhouette Score VS No. Of Clusters

```
from sklearn.metrics import silhouette_samples, silhouette_score

indices = []
scores = []

for kclusters in range(2, 20) :
        kgc = kolkata_grouped_clustering
        kmeans = KMeans(n_clusters = kclusters, init = 'k-means++',
        random_state = 0).fit_predict(kgc)

        score = silhouette_score(kgc, kmeans)

        indices.append(kclusters)
        scores.append(score)

plot(indices, scores)
optimal_value = np.argmax(scores) + 2
```

4.6 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.

```
kclusters = optimal_value

kgc = kolkata_grouped_clustering

kmeans = KMeans(n_clusters = kclusters, init = 'k-means++', random_state = 0) .fit(kgc)
```

5 Results

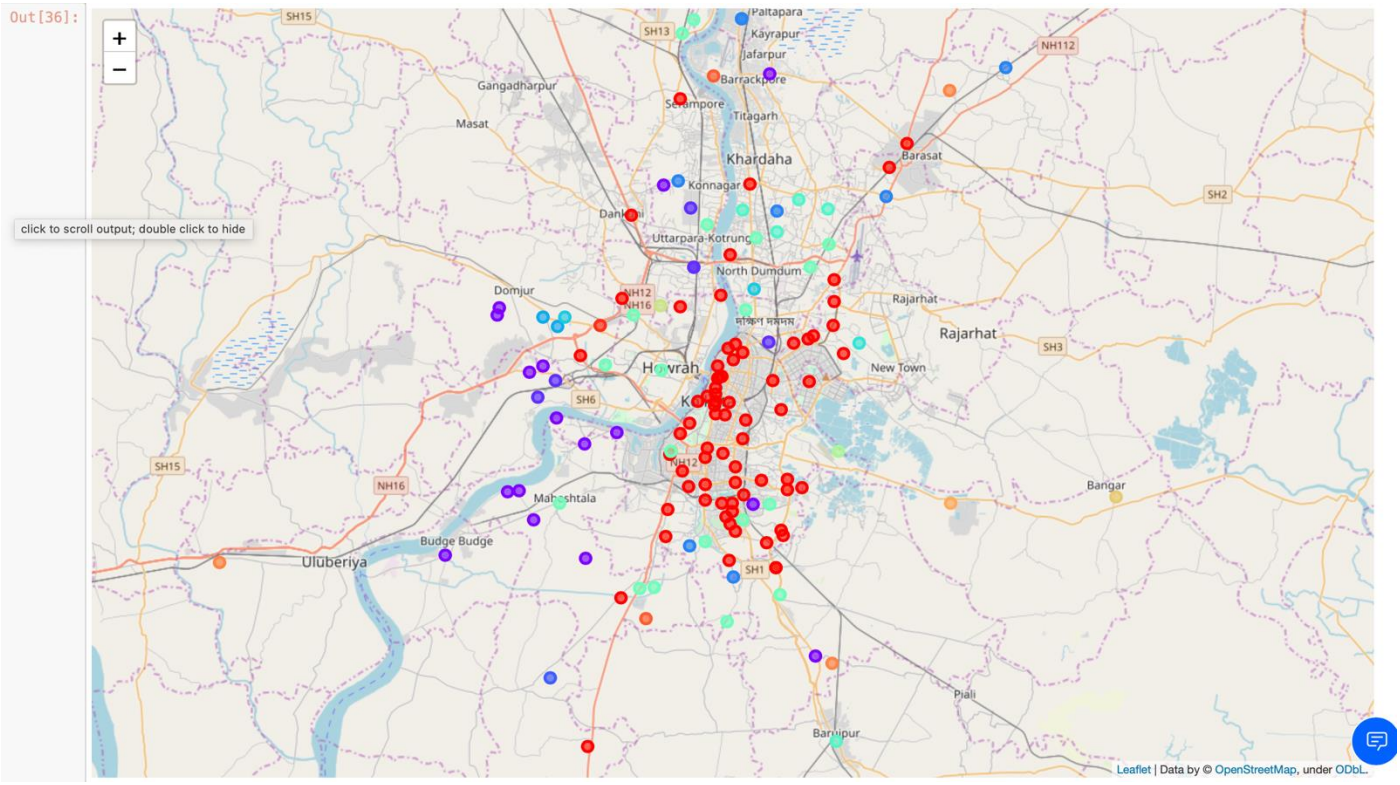
The neighbourhoods are divided into n clusters where n is the number of clusters found using the optimal approach. The clustered neighbourhoods are visualized using different colours so as to make them distinguishable.

```
map_clusters = folium.Map(location=[kol_lat, kol_lng], zoom_start=11)

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]

markers_colors = []
for lat, lon, poi, cluster in zip(kolkata_merged['Latitude'],
kolkata_merged['Longitude'], kolkata_merged['Neighbourhood'],
kolkata_merged['Cluster Labels']):
        label = folium.Popup(str(poi) + ' (Cluster ' + str(cluster + 1) + ')',
        parse_html=True)
        map_clusters.add_child(
        folium.features.CircleMarker( [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7))
```

Out[36]:



6 Discussion

After analysing the various clusters produced by the Machine learning algorithm, **cluster no.18,** is a prime fit to solving the problem of finding a cluster with common venue mentioned.

Out[40]:

	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	10th Most Common Venue
194	Madhyamgram	22.692400	88.465337	4	IT Services	ATM	Pharmacy	Health & Beauty Service	Field	Concert Hall	Convenience Store	Cricket Ground	Currency Exchange	Department Store
195	Maheshtala	22.505590	88.250004	9	Business Service	ATM	Motorcycle Shop	Dumpling Restaurant	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Electronics Store	Diner
196	Mahiari	22.589404	88.238816	1	ATM	Flea Market	Convenience Store	Cricket Ground	Currency Exchange	Department Store	Dessert Shop	Dhaba	Diner	Dumpling Restaurant
197	Makardaha	22.619191	88.238816	5	Flea Market	Women's Store	Convenience Store	Cricket Ground	Currency Exchange	Department Store	Dessert Shop	Dhaba	Diner	Dumpling Restaurant
198	Manikpur, West Bengal	22.469276	88.025303	15	Train Station	Women's Store	Diner	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Electronics Store	Dumpling Restaurant	Dhaba

In the above table we can see the TOP 10 Venues or places to visit in each of the Neighbourhoods of Kolkata are provided.

Kolkata is a big city and this derivation will help tourist in visiting the famous places in their neighbourhood along with providing information on the what's around the corner.

7 Conclusion

This is a self-guide for people about each Neighbourhood to find the most popular places to visit, to eat, for shopping, banks, ATMs etc.
This will also help the boosting the local economy as tourists will go to the famous places around their area of visit. This will also increase the competition among the vendors dealing in similar items as the better their services and ratings the better chance they have in appearing in TOP 10 list.