Coursera Capstone Project - Battle of the Neighbourhoods

Sree Akshaya

Introduction:

Chennai is one of the largest and busiest cities in India making it a great place to set up a business. The issue is, a city as large as itself with thousands of businesses, could be over saturated. Not every neighbourhood might be ideal for all businesses. So, in this project, we seek to find out the best neighbourhoods in Chennai to start up a food related business.

Data:

The neighbourhood data used in the project is taken from Chennai's wikipedia page using the BeautifulSoup library.

```
pd.set_option('max_colwidth', 800)

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

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

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

length = len(mwcg)

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()
df = pd.read_csv('chennai.csv')
df.head()
```

	Neighbourhood
0	Adambakkam
1	Adyar, Chennai
2	Agaram
3	Alandur
4	Alapakkam

The geocoding data, that is, the latitude and longitude of the locations were gathered using Bing's Geocoder API.

```
latlngtable=[]
latf=[]
lngf=[]

for place in df['Neighbourhood'][0:199]:
    latlng = None
    i=0
    while(latlng is None and i<2):
          g = geocoder.bing(place, key = "AkGZ-4V7bLc6jgPQQu0EjhbNdeZFkFO3diqrvXv04SVN-NIG8C6dugh3T-nMsp2q")
          if (len(g) == 1):
             latlng = g.latlng
          else:
             latlng = None
          i+=1
    latlngtable.append(g.latlng)

latlngtable[:5]
```

Latitude	Longitude
12.9919	80.206
13.003	80.2519
10.4555	77.9483
13.0001	80.2005
12.8968	80.1118

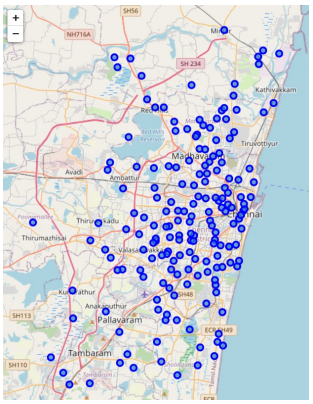
The venue data was collected using the Foursquare API with the neighbourhood and geocoding data as inputs.

```
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
```

Methodology:

Folium was used to create an interactive map of Chennai with the neighbourhoods in question highlighted.

```
chen_lat = 13.0827
chen_lng = 80.2707

map_chennai = folium.Map(location=[chen_lat,chen_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_chennai)

map_chennai
```

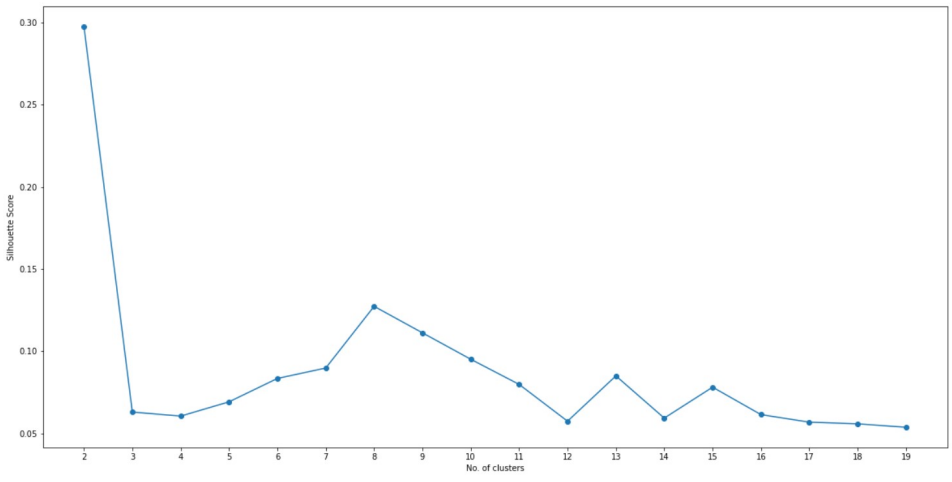


Only the top ten most common venue categories were used since the total number of categories was too much.

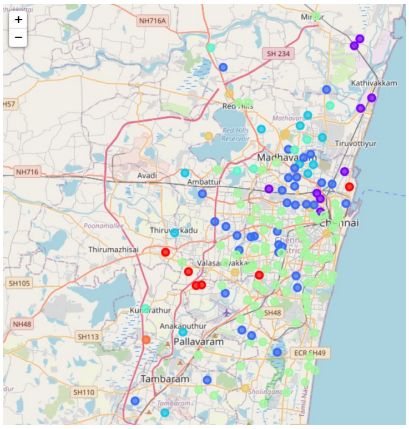
```
num_top_venues = 10
indicators = ['st', 'nd', 'rd']

# 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'] = chennai_grouped['Neighbourhood']

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

neighbourhoods_venues_sorted.head()
```

The Silhouette Score was used to determine how many clusters gave optimal results. Since 2 clusters are too less, 8 clusters, with the next best score was chosen.

```
from sklearn.metrics import silhouette_samples, silhouette_score

indices = []
scores = []

for kclusters in range(2, max_range) :

    kgc = chennai_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)
```



K - Means Clustering Algorithm was used to get the optimal clusters.
In the image we can see 8 different colours with each colour representing a different cluster.




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

Results:

To determine which of the eight clusters was best for starting a food related business, we calculated the number of business venues and food related business venues in each neighbourhood. And then found which cluster had the largest number of neighbourhoods with the number of business venues larger than average (making it a busy place) and the number of food related business venues smaller than average, making it an optimal location and ensuring good business. The cluster that had these properties was Cluster 3, and in that 5 specific neighbourhoods were the best possible choices.

```
food = ['Restaurant', 'Food', 'Bakery', 'Café', 'Joint', 'Diner', 'Bistro', 'Juice', 'Dessert', 'Snack', 'Tea', 'Pizza', 'Sweet', 'Ice Cream']

foodcat=[]
catnames = explore_df['Venue Category'].unique()
for name in catnames:
    i=0
    for foodn in food :
        if(foodn.lower() in name.lower()):
            i+=1
    if(i>0):
        foodcat.append(name)

chennai_grouped_sum['foodsumtotal'] = chennai_grouped_sum[foodcat].sum(axis=1)

chennai_merged_new = chennai_merged
chennai_merged_new['sumtotal'] = 0
chennai_merged_new['foodsumtotal'] = 0
chennai_grouped_sum.index = chennai_merged_new.index

avg_sum = chennai_grouped_sum['sumtotal'].mean()
avg_quo = (chennai_grouped_sum['foodsumtotal']/chennai_grouped_sum['sumtotal']).mean()

dens = []
for val in range(1,9):
    cluster = chennai_merged_new.loc[chennai_merged_new['Cluster Labels'] == (val - 1), chennai_merged_new.columns[np.arange(14,16).tolist()]]
    dens.append(sum((cluster['sumtotal']>avg_sum) & ((cluster['foodsumtotal']/cluster['sumtotal'])<avg_quo))/cluster.shape[0])

cluster3 = chennai_merged_new.loc[chennai_merged_new['Cluster Labels'] == (2), chennai_merged_new.columns[[0] + np.arange(14,16).tolist()]]
cluster3['Ratio'] = cluster3['foodsumtotal']/cluster3['sumtotal']
cluster3.loc[(cluster3['sumtotal']>avg_sum) & ((cluster3['foodsumtotal']/cluster3['sumtotal'])<avg_quo)].sort_values(by='Ratio', ascending = False)
```

	Neighbourhood	sumtotal	foodsumtotal	Ratio
79	Kotturpuram	15	7	0.466667
7	Aminjikarai	18	8	0.444444
77	Kothawal Chavadi	29	12	0.413793
171	Subramania Nagar	17	7	0.411765
121	Pallikaranai	16	4	0.250000

Conclusion:

We have found that for a client who wants to open a food related business in chennai, it is recommended to open it in one of the neighbourhoods in cluster 3, and specifically one of the 5 above mentioned neighbourhoods, for a profitable business venture.