Applied Data Science Capstone

Battle of Neighborhoods : Bengaluru, India

Geospatial Agglomeration with Location Data

ARUN P. R.

June 29, 2020

Contents

List of Figures

Chapter 1

Introduction

This project aims to segregrate the neighborhoods in Bengaluru, India based on their similarities/differences and to make use of this data to interpret the concentration of population in these neighborhoods.

1.1 Background

The distribution of population in neighborhoods has always been a topic of interest to various government/private agencies. The census data is normally used by these agencies. However a comparison between similar neighborhoods is lacking in such studies. This project aims to fill this gap by providing useful insights in to how the neighborhoods are segregrated so that the planning for the above cited activities can be done more effectively. As a representative entity, Bengaluru, India is chosen as the target location.

1.1.1 About Bengaluru, India

Bangalore, officially Bengaluru, is the capital of the Indian state of Karnataka. With a population of over ten million, it is a megacity, the third-most populous city and fifth-most populous urban agglomeration in India. It is widely regarded as the "Silicon Valley of India" and is one of the most productive metro area of India. Bangalore is home to many educational and research institutions in India. Numerous state-owned aerospace and defence organisations are located in the city.

Bangalore was the fastest-growing Indian metropolis after New Delhi between 1991 and 2001, with a growth rate of 38% during the decade. The Bruhat Bengaluru Mahanagara Palike (BBMP, Greater Bangalore Municipal Corporation) formed with 100 wards of the erstwhile Bangalore Mahanagara Palike currently has 198 wards.

A demographically diverse city, Bangalore is the second fastest-growing major metropolis in India. With a population of 8,443,675 in the city and 10,456,000 in the urban agglomeration, up from 8.5 million at the 2011 census, Bangalore is a megacity, and the third-most-populous city in India and the 18th-most-populous city in the world. Bangalore's rapid growth has created several problems relating to traffic congestion and infrastructural obsolescence that the Bangalore Mahanagara Palike has found challenging to address.

The unplanned nature of growth in the city resulted in massive traffic gridlocks that the municipality attempted to ease by constructing a flyover system and by imposing one-way traffic systems. Some of the flyovers and one-ways mitigated the traffic situation moderately but were unable to adequately address the disproportionate growth of city traffic.

1.2 Problem Definition

1.2.1 Business Problem

To segregrate neighborhoods based on location data and along with geospatial and population statistics, arrive at useful insights that can aid different agencices to implement their schemes more effectively.

1.2.2 Description of the problem

The problem consists of following subproblems:

1. Get Data Source for Population/Equivalent for all neighborhoods in Bengaluru, India
2. Get GeoJSON corresponding to neighborhoods in Bengaluru, India
3. Get lattitude, longitude information for all neighborhoods using geocoder
4. Get location data corresponding to all neighborhoods using Foursquare API
5. Clean all data, explore them, extract features
6. Arrive at appropriate methodologies to segregate neighborhoods
7. Segregate neighborhoods using location data
8. Analyse segregated neighborhoods data in conjuction with population data and interpret the results

1.3 Benefits to Stakeholders

The project will be beneficial to various government/private agencies involved in demographic studies, town planning, resource allocation, planning of development projects, etc. by providing useful insights in to how the neighborhoods are segregrated so that the planning for the above cited activities can be done more effectively.

Chapter 2

Data Acquisiton and Cleaning

Details about proposed data sources and how data is proposed to be extracted from them are detailed in this chapter.

2.1 Data Sources

The following are the data sources used in this project:

1. Demographic data of neighborhoods in Bengaluru, India
   - PDF file obtained from website of Karnataka State Election Commission, Ward Wise Voters Data
   - This data in tabular format is available as a pdf file and contains the total number of voters in each neighborhood and is representative of the population.
2. GeoJSON corresponding to neighborhoods in Bengaluru, India
   - BBMP.GeoJSON
   - This dataset is shared under Creative Commons Attribution-ShareAlike 2.5 India license
3. Lattitude, Longitude information for all neighborhoods
   - Geocoder was initially used to get this information.
   - Given that this package can be very unreliable, in this case it was not possible to get the geographical coordinates of the neighborhoods accurately using the Geocoder package, hence it was tweaked from GeoJSON file.
4. Location data corresponding to all neighborhoods
   - Foursquare API is used to get venues and categories for each neighborhood.
   - With neighborhood names and latitude-longitude information, Foursquare API is used to get location data consisting of upto 50 venues within a 2km radius from given geospatial coordinates corresponding to each neighborhood.

2.2 Data Description & Cleaning

Corresponding to each dataset, specific cleaning operations will have to be carried out to make it usable to solve the problem.

For example, WARD WISE VOTERS ABSTRACT contains information useful to this project in columns WARD_NAME and TOTAL whereas all other columns are to be dropped. WARD_NAME is of the format WARD_NO followed by WARD_NAME with a space. WARD_NO needs to be removed from this.

2.2.1 Demographic data of neighborhoods in Bengaluru, India

The voters list summary as of 2010 downloaded from Karnataka Election Commision Website as pdf file. This PDF file consisting of 6 pages was converted to csv file offline and this CSV file was used for the project.

Table has 14 columns including ward no, ward name, male, female and total voters count for selection, addition, deletion and net total. All columns other than ward name and net total are dropped.

The table consists of 198 rows each corresponding to one ward. The ward name has leading ward no information which is stripped. The final dataframe consists of ward name and total voters information.

In [3]:
```
votersDataFrame=pd.read_csv('BBMP_voters.csv')
votersDataFrame.head()
```

Out[3]:

	WARD_NO	WARD_NAME	MALE1	FEMALE1	TOTAL1	MALE2	FEMALE2	TOTAL2	MALE3	FEMALE3	TOTAL3	MALI
0	1	1 Kempegowda ward	14016	12840	26856	1090	890	1980	0	0	0	1510
1	2	2 Chowdeswari ward	12604	11622	24226	1030	857	1887	0	0	0	1363
2	3	3 Atturu	17803	16494	34297	1129	1019	2148	0	0	0	1893
3	4	4 Yelahanka Satellite Town	16201	14974	31175	922	752	1674	0	0	0	1712
4	5	5 Jakkuru	15343	13850	29193	4170	3013	7183	0	0	0	1951

The table contains many unwanted columns. We will select columns of interest to us.

In [4]:
```
votersDataFrame=votersDataFrame[['WARD_NAME','TOTAL']]
votersDataFrame.head()
```

Out[4]:

	WARD_NAME	TOTAL
0	1 Kempegowda ward	28836
1	2 Chowdeswari ward	26113
2	3 Atturu	36445
3	4 Yelahanka Satellite Town	32849
4	5 Jakkuru	36376

The WARD_NAME field is of format {WARD_NO WARD_NAME}. The leading WARD_NO is removed from WARD_NAME field.

In [5]:
```
votersDataFrame['WARD_NAME'] = votersDataFrame['WARD_NAME'].str.lstrip('0123456789.- ')
votersDataFrame.head()
```

Out[5]:

	WARD_NAME	TOTAL
0	Kempegowda ward	28836
1	Chowdeswari ward	26113
2	Atturu	36445
3	Yelahanka Satellite Town	32849
4	Jakkuru	36376

Figure 2.1: BBMP.csv after Data Cleaning

2.2.2 GeoJSON corresponding to neighborhoods in Bengaluru, India

GeoJSON dataset link. This dataset is shared under Creative Commons Attribution-ShareAlike 2.5 India license. This dataset is used to generate chloropleth map of Bengaluru with ward boundaries and total voters from data frame of data source - 1.

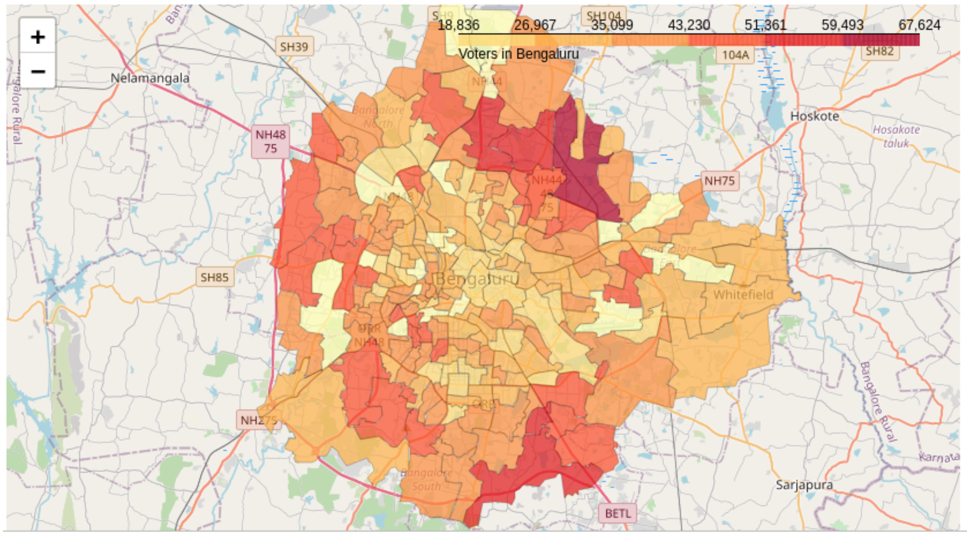

Figure 2.2: Chloropleth of Bengaluru

2.2.3 Lattitude, Longitude information for all neighborhoods

Geocoder was initially used to get this information. The latitude, longitude information was obtained for 186 out of 198 wards. Suitable steps needs to be taken at later stage to address these 12 missing data points.

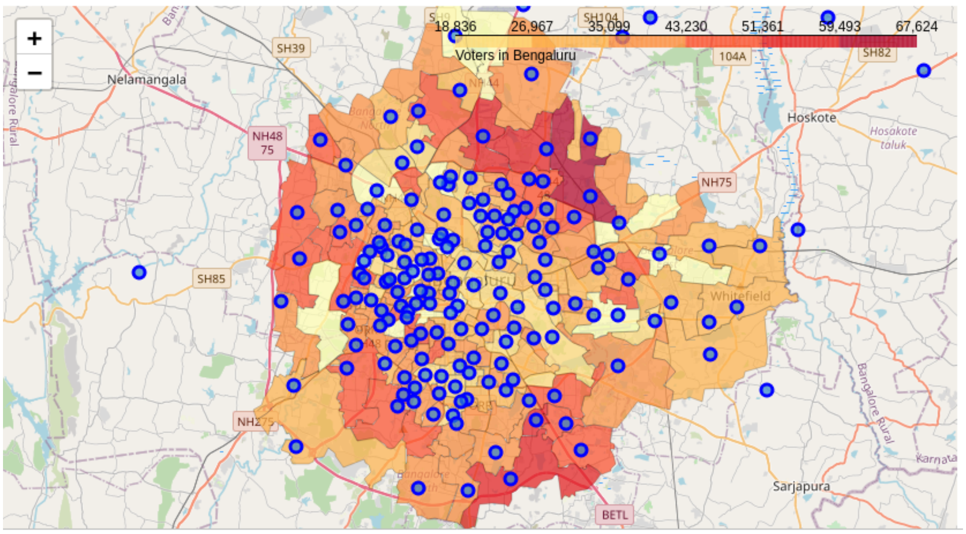

Figure 2.3: Geocoder outputs plotted over Chloropleth

2.2.4 Location data corresponding to all neighborhoods

Foursquare API is used to get venues and categories for each neighborhood. With neighborhood names and latitude-longitude information, Foursquare API is used to get location data consisting of upto 50 venues within a 2km radius from given geospatial coordinates corresponding to each neighborhood.

A total of 7536 venues were returned with the given search criteria. Venues were returned for all 198 wards. A total of 212 unique categories of venues were identified which will form our feature set.

Let's check the size of the resulting dataframe

In [34]:
```
print(neighborhood_venues.shape)
neighborhood_venues.head()
```

```
(7536, 7)
```

Out[34]:

	WARD_NAME	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	A Narayanapura	12.994474	77.672583	Corner House	13.001951	77.661444	Ice Cream Shop
1	A Narayanapura	12.994474	77.672583	Indijoe Restaurant	12.993790	77.661281	Steakhouse
2	A Narayanapura	12.994474	77.672583	Punjab Junction	12.985320	77.673310	Indian Restaurant
3	A Narayanapura	12.994474	77.672583	Auchan	12.992926	77.661842	Convenience Store
4	A Narayanapura	12.994474	77.672583	McDonald's	12.993667	77.666576	Fast Food Restaurant

Let's check how many venues were returned for each neighborhood

In [35]:
```
neighborhood_venues.groupby('WARD_NAME').count().head()
```

Out[35]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
WARD_NAME						
A Narayanapura	49	49	49	49	49	49
Adugodi	50	50	50	50	50	50
Agaram	50	50	50	50	50	50
Agrahara Dasarahalli	49	49	49	49	49	49
Anjanapura	7	7	7	7	7	7

Let's find out how many unique categories can be curated from all the returned venues

In [36]:
```
print('There are {} uniques categories.'.format(len(neighborhood_venues['Venue Category'].unique
())))
```

```
There are 212 uniques categories.
```

Figure 2.4: Foursquare Data for Venues

Chapter 3

Analytic Approach

3.1 Exploratory Data Analysis

3.1.1 Analysis of Lattitude, Longitude data for all neighborhoods

Geocoder was initially used to get this information. The latitude, longitude information was obtained for 186 out of 198 wards. Suitable steps needs to be taken at later stage to address these 12 missing data points.

The chloropleth with the available data points were plotted and it was noticed that the points were not accurate and there were points far from the chloropleth boundary which may be due to similarly named places outside Bengaluru. So in addition to missing data, the given data source was giving inaccurate results in many cases.

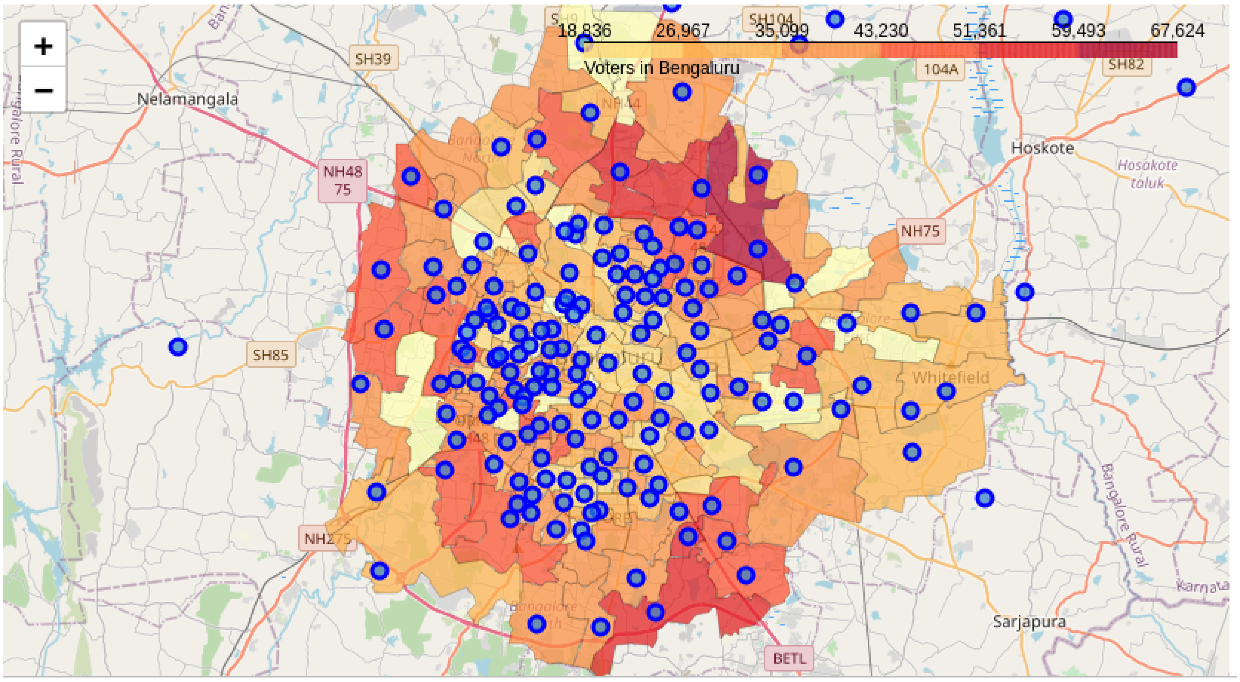

Figure 3.1: Geocoder outputs plotted over Chloropleth

The situation was analysed with box plots and it was observed that this data is having many outliers.

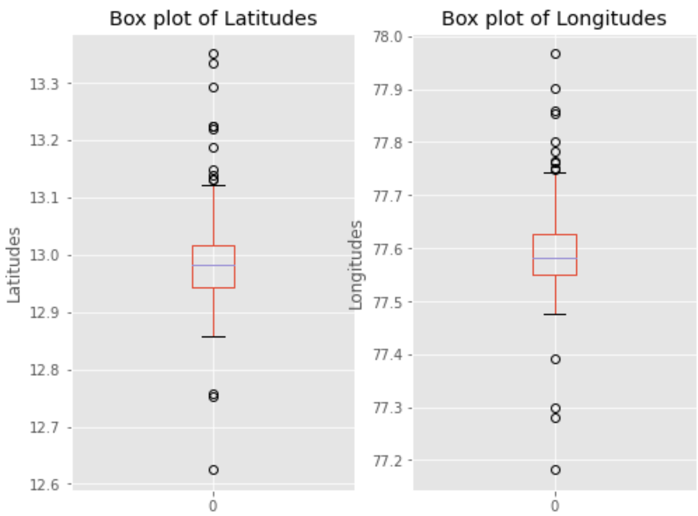

Figure 3.2: Box plot for Geocoder outputs - many outliers far away from quartiles

Given that this package can be very unreliable, in this case it was not possible to get the geographical coordinates of the neighborhoods accurately using the Geocoder package, hence it was tweaked from GeoJSON file.

GeoJSON file (Data Source 2) file consists of feature - properties which includes ward name, latitude, longitude which can serve our purpose given the fact that geocoder data pproved to be unusable. The GeoJSON file was parsed to obtain the required information. The latitude, longitude information was obtained for all 198 wards. However on merging by ward names, it was observed that only 181 wards are getting parsed correctly.

Using outer join, the remaining ward names were seen and it was noticed that there are slight mismatches in names between the data sets for 17 wards.

In [19]:
```
votersDataFrame.merge(df_latlong,on='WARD_NAME',sort=True).shape#inplace=True not done as we hav
e to merge all rows
```

Out[19]: (181, 4)

It can be seen that only 181 out of 198 ward locations were decoded correctly. Lets analyse the wards that could not be decoded using outer join on data frames.

In [20]:
```
is_NaN = votersDataFrame.merge(df_latlong,on='WARD_NAME',how='outer').isnull()
row_has_NaN = is_NaN.any(axis=1)
temp=votersDataFrame.merge(df_latlong,on='WARD_NAME',how='outer')[row_has_NaN]
temp.reset_index(inplace=True)
temp
```

Out[20]:

	index	WARD_NAME	TOTAL	latitude	longitude
0	0	Kempegowda ward	28836.0	NaN	NaN
1	1	Chowdeswari ward	26113.0	NaN	NaN
2	17	Radhakrishna Temple ward	35763.0	NaN	NaN
3	37	HMT ward	32536.0	NaN	NaN
4	50	Vijinapura	42897.0	NaN	NaN
5	51	K R Pura	36877.0	NaN	NaN
6	64	Kadu Malleshwara ward	30083.0	NaN	NaN
7	76	Dattatreya Temple ward	37717.0	NaN	NaN
8	81	Garudachar Palya	31975.0	NaN	NaN
9	86	HAL Airport ward	41792.0	NaN	NaN
10	101	Vrisabhavathi	34317.0	NaN	NaN
11	107	Sriramamandir Ward	29659.0	NaN	NaN
12	118	Dharmaraya Swamy Temple Ward	27070.0	NaN	NaN
13	121	Kempapura	31933.0	NaN	NaN
14	128	Jnana Bharathi Ward	49763.0	NaN	NaN
15	139	CHAMARAJ PET	29205.0	NaN	NaN
16	156	Gali Anjaneya Temple ward	30805.0	NaN	NaN
17	198	Chowdeswari Ward	NaN	13.121709	77.580422
18	199	Vijnanapura	NaN	13.006063	77.669565
19	200	HAL Airport	NaN	12.956537	77.671502
20	201	HMT Ward	NaN	13.031905	77.531705

Figure 3.3: Data Parsing Error in Ward Names

A dictionary was prepared mapping these 17 ward names between two data sets and the data was updated and merged successfully.

```
In [21]: temp1=sorted(temp[['WARD_NAME']][0:17].values.tolist())
         temp2=sorted(temp[['WARD_NAME']][17:34].values.tolist())
         dictionary =dict()
         for i in range (0,len(temp1)):
             dictionary[str(temp2[i][0])]=str(temp1[i][0])
         dictionary

Out[21]: {'Chamrajapet': 'CHAMARAJ PET',
          'Chowdeswari Ward': 'Chowdeswari ward',
          'Dattatreya Temple': 'Dattatreya Temple ward',
          'Dharmaraya Swamy Temple': 'Dharmaraya Swamy Temple Ward',
          'Gali Anjenaya Temple ward': 'Gali Anjaneya Temple ward',
          'Garudachar Playa': 'Garudachar Palya',
          'HAL Airport': 'HAL Airport ward',
          'HMT Ward': 'HMT ward',
          'Jnana Bharathi ward': 'Jnana Bharathi Ward',
          'K R Puram': 'K R Pura',
          'Kadu Malleshwar Ward': 'Kadu Malleshwara ward',
          'Kempapura Agrahara': 'Kempapura',
          'Kempegowda Ward': 'Kempegowda ward',
          'Radhakrishna Temple Ward': 'Radhakrishna Temple ward',
          'Sriramamandir': 'Sriramamandir Ward',
          'Vijnanapura': 'Vijinapura',
          'Vrisabhavathi Nagar': 'Vrisabhavathi'}
```

Now replace these WARD_NAMES in latlong with corresponding names in voters

```
In [22]: for i in dictionary:
             df_latlong[['WARD_NAME']]=df_latlong['WARD_NAME'].str.replace("{}".format(i),dictionary[i])
```

Now check if all ward names are decoded correctly.

```
In [23]: votersDataFrame.merge(df_latlong,on='WARD_NAME',sort=True).shape#inplace=True not done as we hav
         e to merge all rows

Out[23]: (198, 4)
```

Create our merged data frame containing voter and latlong details.

```
In [24]: merged_df=votersDataFrame.merge(df_latlong,on='WARD_NAME',sort=True)
         merged_df.head()
```

Out[24]:

	WARD_NAME	TOTAL	latitude	longitude
0	A Narayanapura	31375	12.994474	77.672583
1	Adugodi	26320	12.943239	77.613079
2	Agaram	24577	12.944263	77.639047
3	Agrahara Dasarahalli	27453	12.980497	77.541535
4	Anjanapura	36226	12.859588	77.563286

Figure 3.4: Data Update with Dictionary mapping for 17 ward names

The chloropleth with the data points were plotted and it was noticed to be normal compared to the earlier rendition.

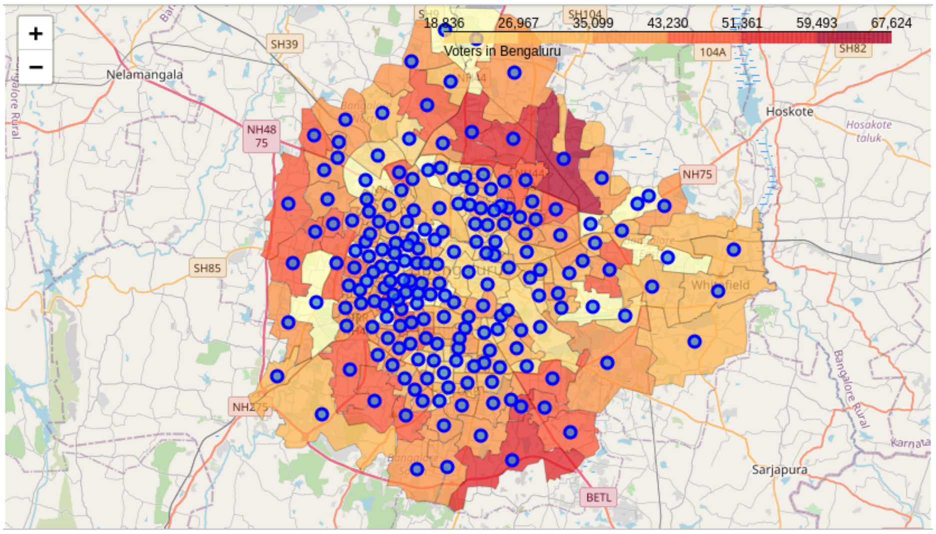

Figure 3.5: GeoJSON outputs plotted over Chloropleth

The situation was analysed with box plots and this data is having only a few outliers corresponding to the outer wards.

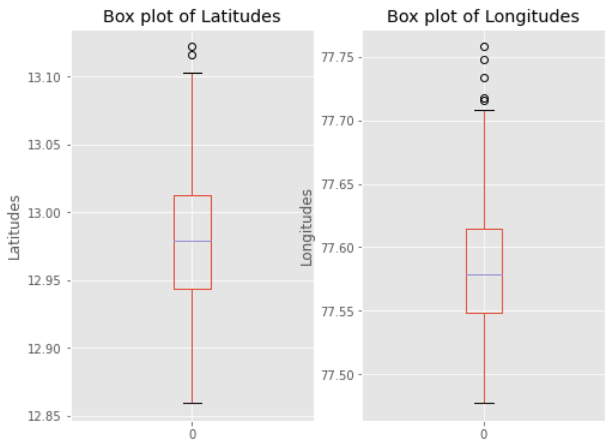

Figure 3.6: Box plot for GeoJSON outputs - few outliers

Chapter 4

Methodology

The neighborhood data is to be segregated using the feature set venue categories provided by Foursquare API. A clustering algorithm needs to be chosen for this and k-Means Clustering Algorithm is used in this project. The optimum number of clusters are to be determined and k-Means clustering is carried out. The chloropleth map of Bengaluru with voters data is prepared and the neighborhoods segregated by clusters are overplotted on this. This data is to be analysed to arrive at insights in to the relation between the clusters and population.

4.1 Feature Extraction & Normalization

Feature selection to be carried out before segregating neighborhoods so that we get a good segregation. The features selected in this case are venue categories derived from Foursqare API.

The 212 features were encoded by 7536 venues using onehot encoding. The venues were grouped by ward names, normalizing the values in the go, leaving us with 198 rows (one for each ward) with 212 features each.

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [39]:
```
neighborhood_grouped = neighborhood_onehot.groupby('WARD_NAME').mean().reset_index()
neighborhood_grouped.head()
```

Out[39]:

	WARD_NAME	ATM	Afghan Restaurant	Airport	Airport Terminal	American Restaurant	Andhra Restaurant	Arcade	Art Gallery	Art Museum	Arts & Crafts Store	Asi Res
0	A Narayanapura	0.000000	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0	0.0	0.02
1	Adugodi	0.000000	0.0	0.0	0.0	0.0	0.02	0.02	0.0	0.0	0.0	0.00
2	Agaram	0.000000	0.0	0.0	0.0	0.0	0.00	0.02	0.0	0.0	0.0	0.02
3	Agrahara Dasarahalli	0.000000	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0	0.0	0.02
4	Anjanapura	0.285714	0.0	0.0	0.0	0.0	0.00	0.00	0.0	0.0	0.0	0.00

Let's confirm the new size

In [40]:
```
neighborhood_grouped.shape
```

Out[40]: (198, 213)

Figure 4.1: Normalized Features

The data frame with top ten venues per ward was formed for later analysis.

In [43]:
```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['WARD_NAME']
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 dataframeStatistics
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['WARD_NAME'] = neighborhood_grouped['WARD_NAME']

for ind in np.arange(neighborhood_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(neighborhood_grouped.i
loc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out[43]:

	WARD_NAME	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 Mos Commo Venue
0	A Narayanapura	Indian Restaurant	Café	Coffee Shop	Gym	Pizza Place	Fast Food Restaurant	Department Store	Bus Station	Ice Crea Shop
1	Adugodi	Indian Restaurant	Café	Clothing Store	Ice Cream Shop	Breakfast Spot	Lounge	Bakery	Dessert Shop	Chinese Restaura
2	Agaram	Indian Restaurant	Ice Cream Shop	Café	Pizza Place	Gym / Fitness Center	Juice Bar	Brewery	Bakery	Middle Eastern Restaura
3	Agrahara Dasarahalli	Indian Restaurant	Fast Food Restaurant	Ice Cream Shop	Café	Pizza Place	Bakery	Coffee Shop	Department Store	Chinese Restaura
4	Anjanapura	ATM	Pool	Flower Shop	Supermarket	Coffee Shop	Convenience Store	Creperie	Dumpling Restaurant	Field

Figure 4.2: Top 10 venues by ward

4.2 K-Means Clustering

4.2.1 Finding optimum k

Elbow method was used to arrive at an optimum value of k for k-means clustering. Based on elbow method a cluster size of 3(based on inertia) or 4(based on distortion) can be chosen. k=4 is chosen as cluster size.

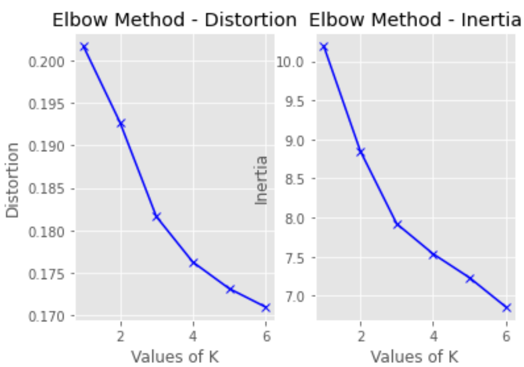

Figure 4.3: Elbow Method to determine k

Chapter 5

Results

5.1 Clusters and Top 10 Venues by ward

After successful clustering, Cluster sizes are 54,8,70,66 with k=4.

```
In [48]: # set number of clusters
         kclusters = 4

         # run k-means clustering
         kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(neighborhood_grouped_clustering)

         # check cluster labels generated for each row in the dataframe
         kmeans.labels_[:]

Out[48]: array([0, 3, 3, 2, 1, 0, 3, 0, 2, 0, 3, 2, 3, 2, 0, 2, 0, 0, 1, 0, 0, 3,
                3, 2, 2, 2, 2, 0, 2, 2, 2, 0, 0, 3, 2, 3, 2, 0, 2, 3, 2, 1, 2, 2,
                3, 2, 3, 0, 2, 2, 2, 0, 3, 3, 0, 3, 2, 3, 2, 3, 3, 0, 3, 3, 0, 2,
                2, 1, 3, 1, 3, 2, 3, 0, 0, 3, 0, 3, 3, 0, 3, 2, 3, 3, 0, 2, 2, 3,
                3, 2, 3, 2, 0, 2, 3, 3, 2, 2, 3, 2, 0, 0, 2, 2, 0, 0, 0, 3, 3, 3,
                0, 0, 2, 2, 1, 3, 0, 2, 3, 3, 0, 3, 0, 2, 3, 2, 2, 0, 2, 0, 0, 2,
                3, 0, 0, 0, 0, 3, 3, 2, 0, 0, 3, 3, 2, 2, 3, 2, 1, 3, 3, 0, 1, 2,
                0, 2, 2, 3, 2, 3, 3, 3, 0, 0, 3, 3, 2, 3, 2, 2, 3, 2, 2, 2, 3, 3,
                3, 2, 0, 0, 3, 2, 2, 3, 2, 0, 2, 3, 2, 0, 2, 2, 0, 0, 2, 2, 0, 3],
               dtype=int32)

In [49]: np.unique(kmeans.labels_, return_counts=True)

Out[49]: (array([0, 1, 2, 3], dtype=int32), array([54,  8, 70, 66]))
```

Figure 5.1: K-Means Clustering

The cluster label was merged with the top 10 venues per ward data frame for further analysis.

	WARD_NAME	TOTAL	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 Mos Commo Venue
0	A Narayanapura	31375	12.994474	77.672583	0	Indian Restaurant	Café	Coffee Shop	Gym	Pizza Place	Fast Foo Restaur
1	Adugodi	26320	12.943239	77.613079	3	Indian Restaurant	Café	Clothing Store	Ice Cream Shop	Breakfast Spot	Lounge
2	Agaram	24577	12.944263	77.639047	3	Indian Restaurant	Ice Cream Shop	Café	Pizza Place	Gym / Fitness Center	Juice Ba
3	Agrahara Dasarahalli	27453	12.980497	77.541535	2	Indian Restaurant	Fast Food Restaurant	Ice Cream Shop	Café	Pizza Place	Bakery
4	Anjanapura	36226	12.859588	77.563286	1	ATM	Pool	Flower Shop	Supermarket	Coffee Shop	Conveni Store

Figure 5.2: Top 10 venues per ward with clusters

Chapter 6

Discussion

The results are promising in that our initial assumption that clusters based on locations, venue categories in this case, is dependant on population. This chapter analyses the results obtained above.

From the cholouropleth, Clusters 1 and 2 are concentrated towards the city, whereas clusters 3 and 4 are formed by wards to the exterior of the city. Clusters 1 and 2 include different wards covering different voter counts whereas clusters 3 and 4 mostly include wards where the no of voters is less.

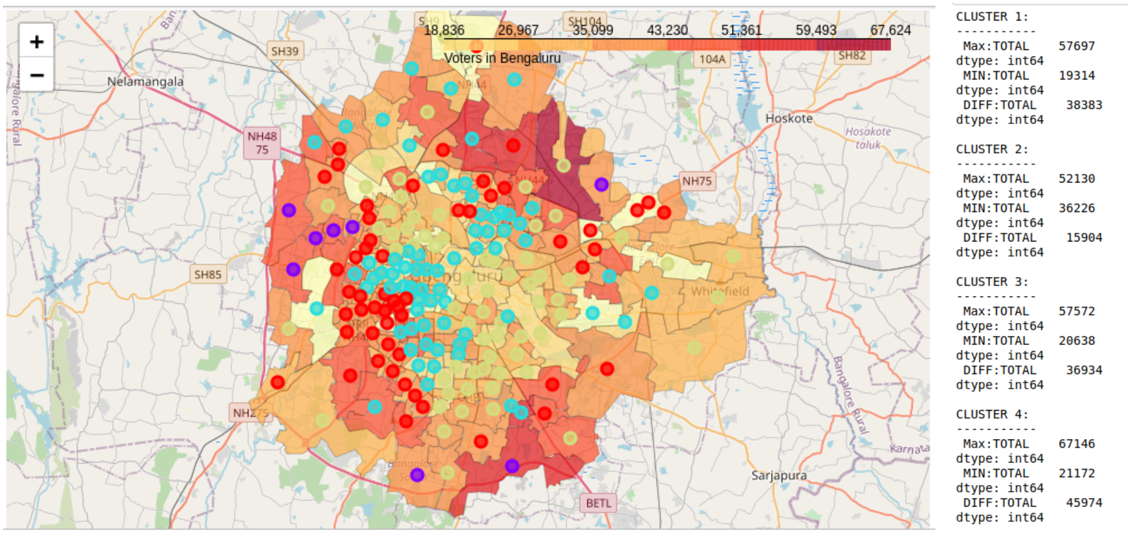

Figure 6.1: Clusters in Chloropleth and Cluster Statistics by min, max voter count

Thus the project results validate our initial premises.

Chapter 7

Conclusion & Future Directions

This chapter elucidates the conclusions of this project and enlightens the way to move forward with further work in the selected areas.

7.1 Conclusion

- Inner wards with more population are clustered together based on the feature set which includes venue categories. This is intuitive since more locations/venues will be available towards the centre of city as well as places where population is more.
- Outer wards with lesser population are clustered together based on the feature set which includes venue categories. This is intuitive since lesser locations/venues only will be available towards the outskirts of city as well as places where population is less.
- Thus the results validate our initial premise that the population data in conjuction with location data will give us insights in to how clusters are formed around cities.

7.2 Future Directions

- The study was based on Bengaluru, India taken as a representative entity. The study can be extended to more cities.
- More variables can be brought in to the feature set to get more insights in to the data.
- Location data can be made more specific to search for business avenues in different clusters.
- ...And keep imagining, even the sky is not the limit.