Coursera Capstone

IBM Applied Data Science Capstone

Finding a Neighborhood to move-in in a new City

Bangalore, India

By: Prashanth Ravi Shankar

June 2020

1. Introduction

A migrant worker is a person who either migrates within their home country or outside it to pursue work. Migrant workers usually do not have the intention to stay permanently in the country or region in which they work. [Wikipedia] Millions of skilled and un-skilled workers migrated to another city every year. When a skilled migrant worker moves to another city in a country. He/She finds it difficult to find a safe and a neighborhood with all basic and common amenities in them. This is a hypothetical clustering analysis of such migrant worker who is planning to move to a new city and need to find a neighborhood to move-in and eventually settle down there.

2. Business Problem

The Objective of this capstone project is to select the optimal location close by to office and at the same time finding a neighborhood that has all the common amenities like parks, hotels, etc. With the help of data science methodology and machine leaning technique like clustering analysis, this projects aims to solve a problem that is faced by every migrant worker faces when He/She moves to a new city. The major question is, which is the best neighborhood to stay?

3. Data

The data for this project has been gathered from multiple sources and processed for clustering.

The Bangalore neighborhood data is not readily available, the data is extracted from Wikipedia page using BeautifulSoup library for Python and scraped the required data from the webpage. This data does not have latitude and longitude coordinates with it, so Google's Geocoding API is used to gather latitude and longitude information.

After that, we will use Foursquare API to get the venue data for those neighborhoods. Foursquare has one of the largest database of 105MM places and is used by over 125,000 developers. Foursquare API will provide many categories of the venue data and we use the available information for each neighborhood for this clustering.

4. Methodology

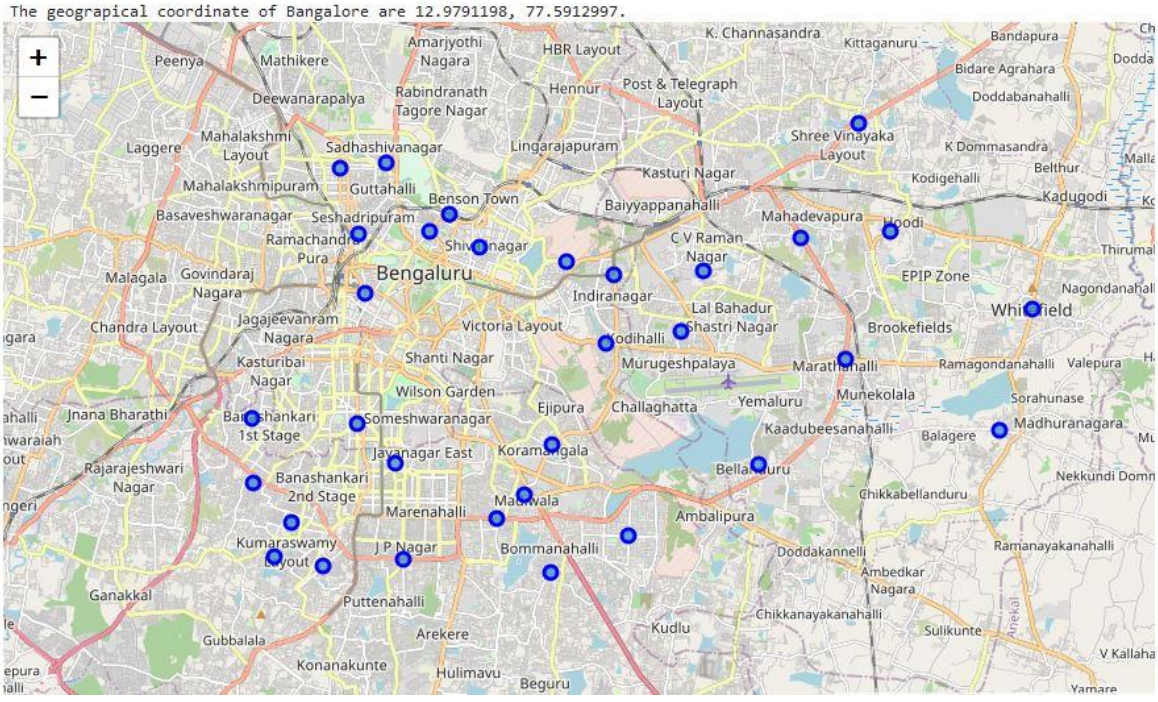
A thorough analysis has been made to ensure the clustering is done properly. Right from Data gathering, Data pre-processing, data analysis, and clustering.

Folium

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the leaflet.js library. This library is used throughout this project to create interactive maps in which we plot the neighborhoods.

```
address = 'Bangalore, Karnataka'

geolocator = Nominatim(user_agent="in_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Bangalore are {}, {}.'.format(latitude, longitude))

# create map of New York using latitude and longitude values
blore = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers to map
for lat, lng, borough, neighborhood in zip(blore_brough['latitude'], blore_brough['longitude'],
                                           blore_brough['Brough'], blore_brough['Neighbourhood']):
    label = '{}, {}'.format(neighborhood, borough)
    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(blore)

blore
```

```
The geograpical coordinate of Bangalore are 12.9791198, 77.5912997.
```



OneHot Encoding

OneHot encoding transposes a categorical column into multiple column based on the number of category and assign binary values for each column. This helps machine learning algorithms to do a better job at prediction. Here, all the `Venue Category values` are onehot encoded.

```
# one hot encoding
blore_onehot = pd.get_dummies(blore_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
blore_onehot['Neighbourhood'] = blore_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [blore_onehot.columns[-1]] + list(blore_onehot.columns[:-1])
blore_onehot = blore_onehot[fixed_columns]

blore_onehot.head()
```

Top 25 venues

Due to high variety of venues, only the top 25 venues for each neighborhood is selected for data exploration. This data is then used to perform k-means clustering.

```
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 = 25

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
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighbourhood'] = blore_grouped['Neighbourhood']

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

neighborhoods_venues_sorted.head()
```

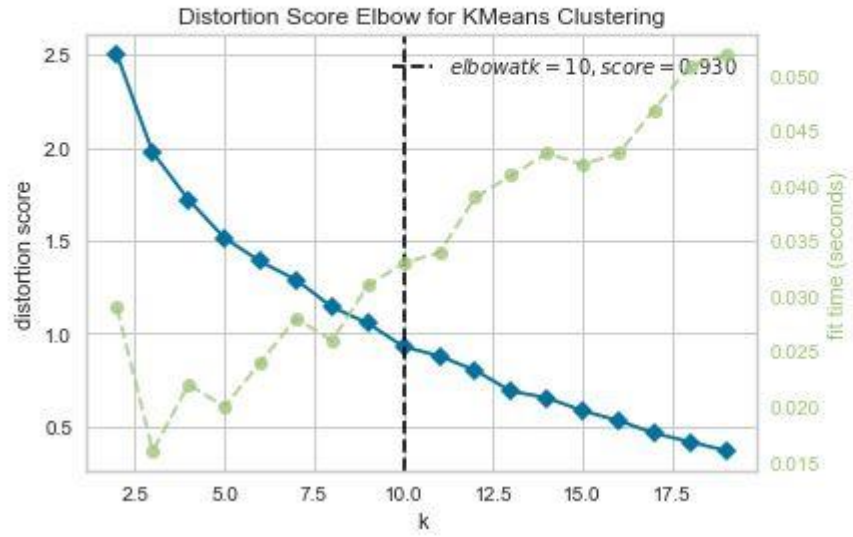
Cluster Determination:

To perform clustering analysis, we need to determine the number of cluster to be created with the data. Elbow method is used for this purpose.

In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use. [Wikipedia]

Based on the elbow method run on 20 k values, the optimal cluster size is determined as 10

```
blore_grouped_clustering = blore_grouped.drop('Neighbourhood', 1)

kelbow_visualizer(KMeans(random_state=0), blore_grouped_clustering, k=(2,20))
```



K-Means Clustering

The Grouped Venue data is trained using k-means clustering algorithm from scikt-learn python package to get the desired clusters to base the analysis on.

```
# set number of clusters
kclusters = 10

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

5. Results

The neighborhood is divided into 10 clusters where 10 is the cluster count found using elbow method. The clusters are visualized using folium package with assigning different colors for each clusters.

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
rainbow = ['#D66853', '#7D4E57', '#364156','#212D40','#11151C', '#003265', '#0061A7', '#FDAD01', '#FC611C', '#CD3952']

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(blore_merged['latitude'], blore_merged['longitude'],
                                  blore_merged['Neighbourhood'], blore_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

folium.Marker([12.937162,77.695193], popup='Office').add_to(map_clusters)

map_clusters
```

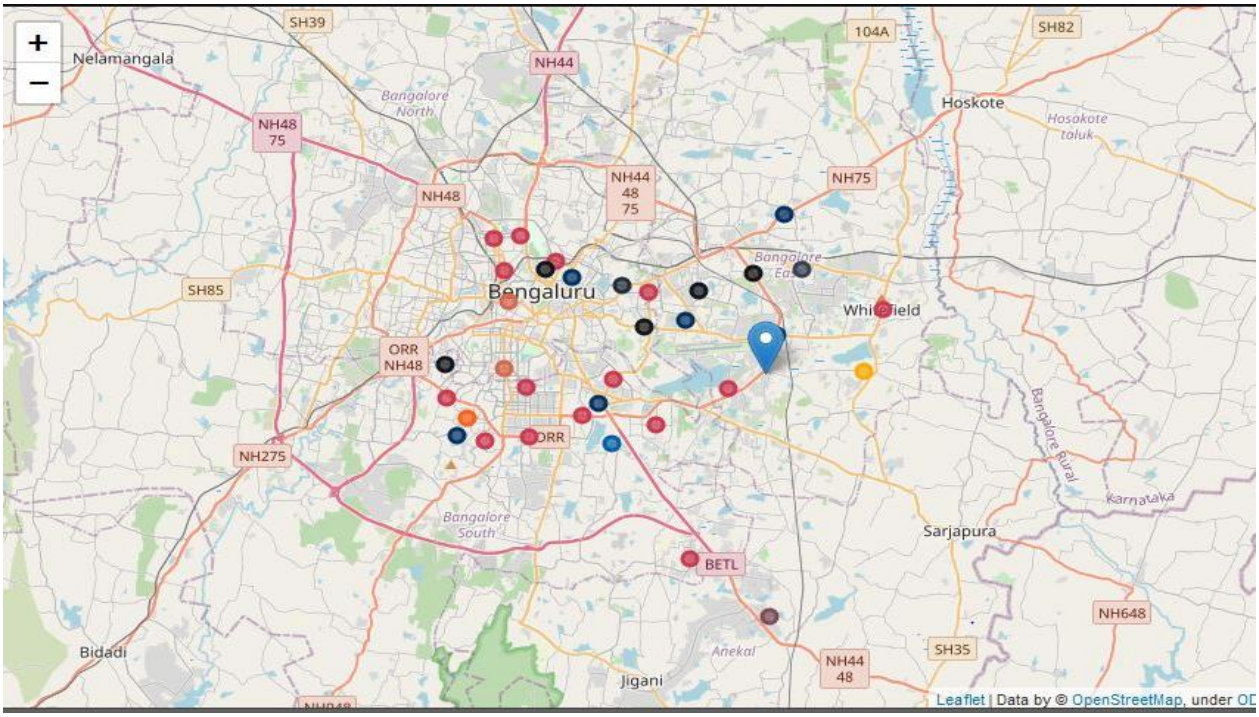


Figure 1 Neighborhoods of Bangalore (Clustered)

6. Discussion

After analyzing the various clusters produced by k-means algorithm, clusters 0,1,5,6 are considered to be the best clusters to move-in after considering the real estate amounts.

Areas like, Bellandur, Mahadevapura, Domlur, Koramangala, Indranagar, Basavanagudi are popular areas with all the common amenities are available in it.

7. Limitation and Suggestions for Future research

In this Project, we considered only the nearby venues for each neighborhood. There are other factors like are population, real estate rents, popular internet connectivity etc. To the best knowledge of researchers, these data were not available at neighborhood level plus Foursquare API has a limit restricting in their response.