CAPSTONE PROJECT

FINDING A NEIGHBORHOOD TO MOVE-IN IN A NEW CITY; BANGALORE, INDIA

PRASHANTH RAVI SHANKAR

JUNE 2020

INTRODUCTION

- A migrant worker is a person who either migrates within their home country or outside it to pursue work.
- 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.

BUSINESS PROBLEM

- Objective is to select the optimal location close by to office
- 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.
- The major question is, which is the best neighborhood to stay?

DATA

- Neighborhood data from Wikipedia.
  - Extracted using BeautifulSoup python package
- Latitude and Longitude data from Google's Geocode API
  - Gathered using resfull api and requests python package
- Gathering venue information from Foursqure API
  - Available venue information are captured using foursquare API
  - Venue information are gathered at neighborhood level.

METHODOLOGY

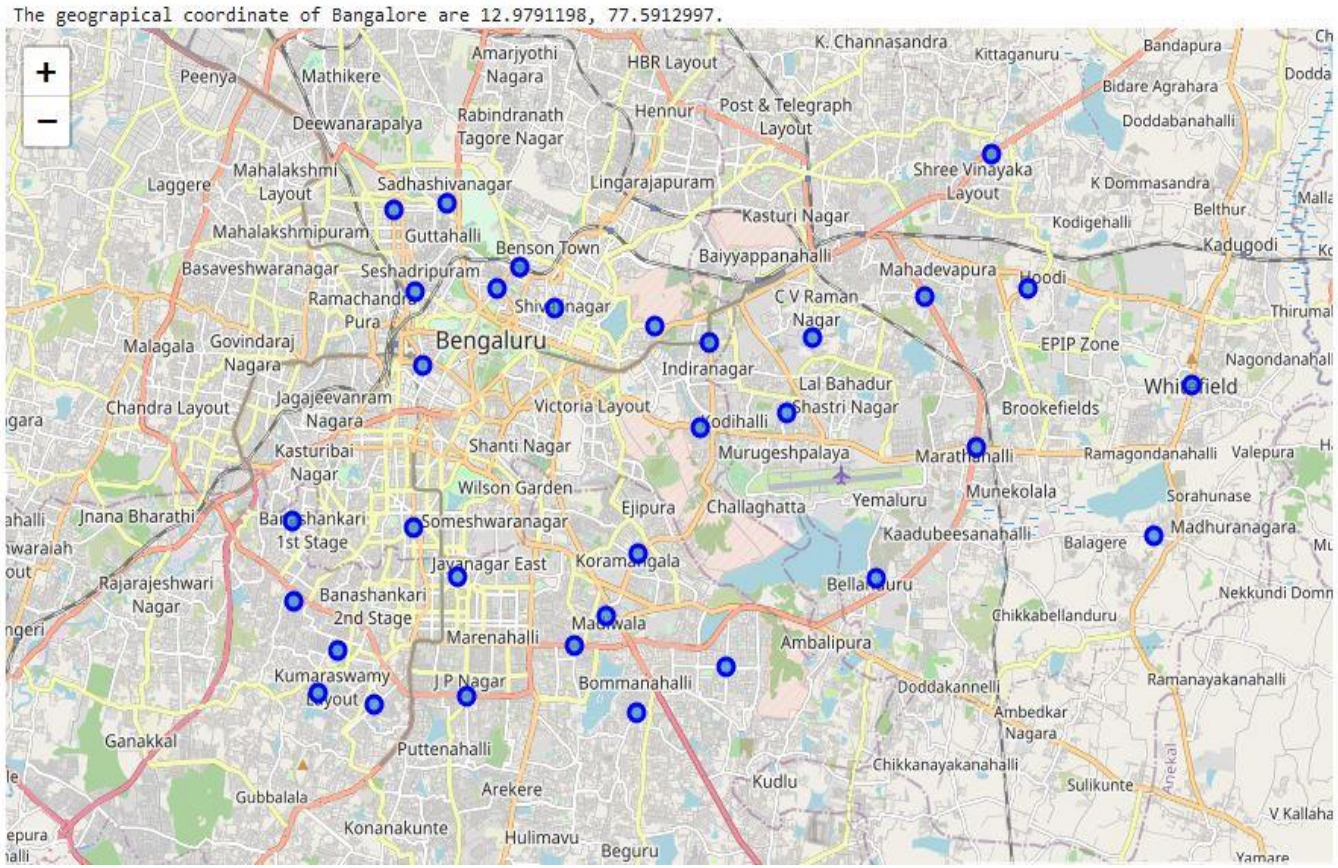
- Folium

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the leaflet.js library.

```
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
```

METHODOLOGY

- Map of Bangalore with neighborhoods are mapped onto it.



METHODOLOGY

- OneHot Encoding
- OneHot encoding transposes a categorical column into multiple column based on the number of category and assign binary values for each column.
- 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()
```

METHODOLOGY

- 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.

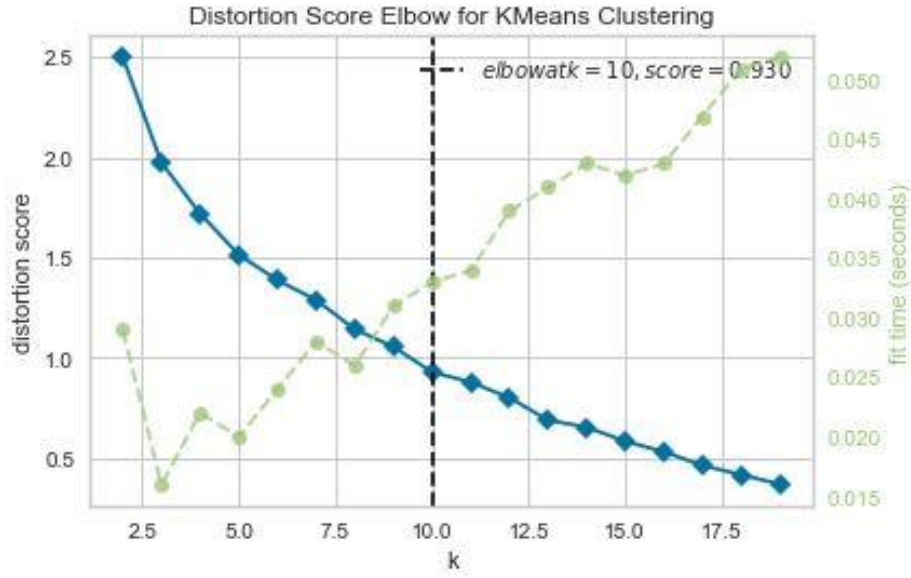
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    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()
```

METHODOLOGY

- **Cluster Determination**
- Elbow method is used to determine optimal cluster
- 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.

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

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



METHODOLOGY

- **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.

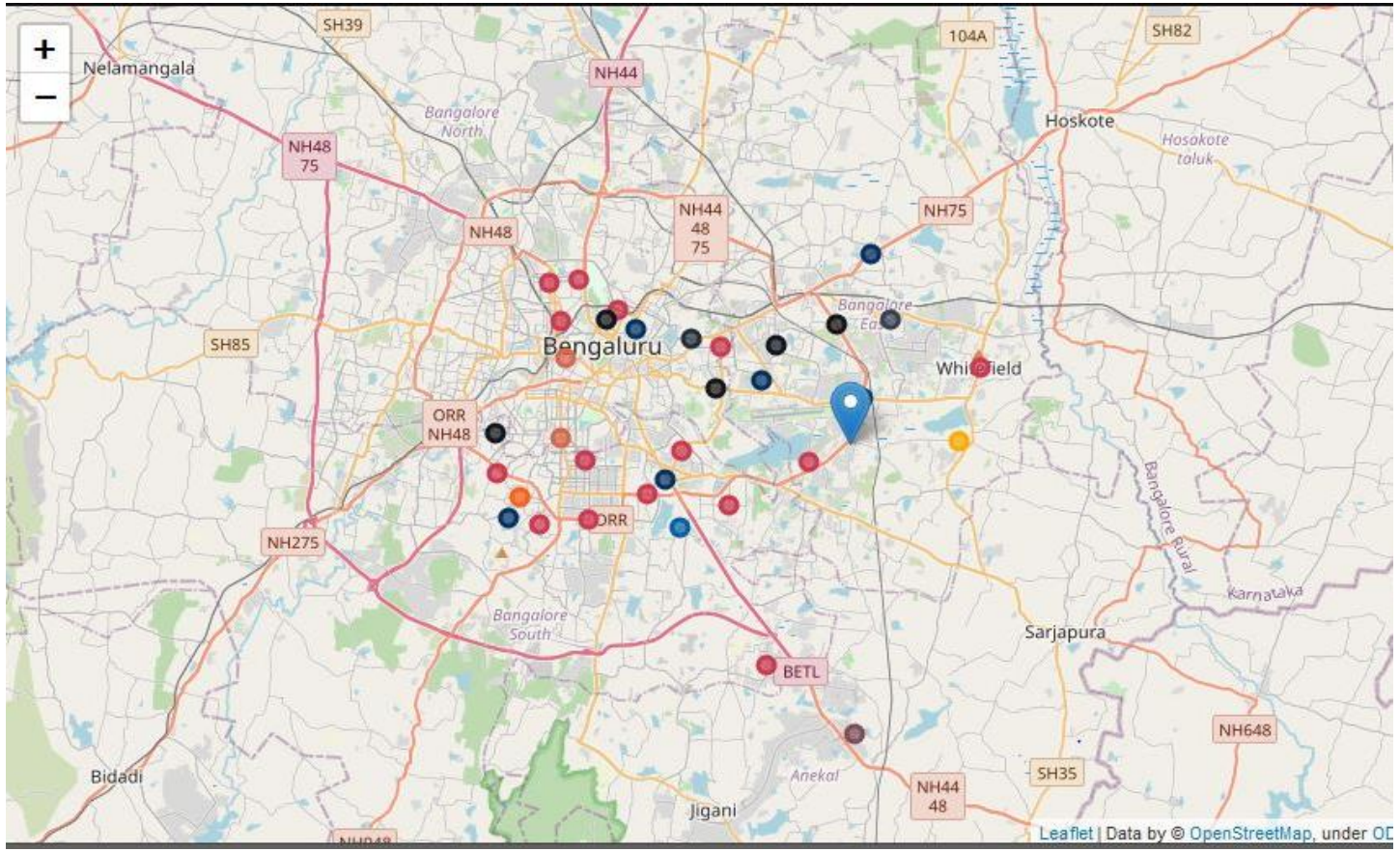
```
# set number of clusters
kclusters = 10

# 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]
```

RESULTS

- The neighborhood is divided into 10 clusters where 10 is the cluster count found using elbow method.

```
# 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
```

RESULTS


SH39
Nelamangala
NH44
104A
SH82
Bangalore North
Hoskote
Hosakote taluk
NH48 75
NH48
NH44 48 75
NH75
Bangalore East
SH85
Bengaluru
Whitefield
ORR NH48
ORR
NH275
Bangalore South
Bangalore Rural
Karnataka
Sarjapura
BETL
NH648
Bidadi
Anekal
SH35
NH44 48
Jigani
Leaflet | Data by © OpenStreetMap, under OD

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

LIMITATION AND SUGGESTIONS FOR FUTURE RESEARCH

- Considered only the nearby venues for each neighborhood.
- Other factors like are population, real estate rents, popular internet connectivity etc. can be included.
- Plus Foursquare API has a limit restricting in their response.

Q&A

Any Questions?

Thank You!