IBM – Coursera
Data Science Specialization

Capstone project - Final report

Correlation between a neighborhood real estate price and its surrounding venues

Suraj Tiwari - March, 2020

Table of Contents

01. Introduction

Kolkata Railway is a suburban rail system serving the suburbs surrounding the city of Kolkata in West Bengal, India. Railways such as these are important and heavily used infrastructure in India. It is the largest suburban railway network in India by track length and number of stations. It has 393 stations and a track length of 1,332 km. The suburban railway operates 1497 EMU services carrying 3.5 million (35 lakhs) people daily. It runs from 4 a.m to 2 a.m in the night.

As per the report on Kolkata Suburban Railway, Eastern Railways and South Eastern Railways have published statistics on passenger services. In the period of 2010–11, the average train services per day were 1275 and average passengers capacity per rake were 6207 and In 2014–15, the average train services were 1511 and average passenger capacity per rake was 4141. which it concludes from last five years, there was an increase of 3% in average services per day and deduction of 8% in average passenger capacity per rake and the no. of passengers moved in period 2013–14 was 115 crores and in the period of 2014–15 was 112 crores which means a reduction of 3 per cent in total trips.

In terms of the fare prices, as per the 2020 Indian Annual Budget, the railway increased the ticket fare by 10 paise per kilometre, although the railway ministry has hiked it by 8 paise per kilometre erlier in 2014. The number of slabs has also been reduced to four—Rs.5 (7.2 US cents), Rs.10 (14 US cents), Rs.15 (22 US cents) and Rs.20 (29 US cents)—from the eight slabs earlier before 2014 railways budget. Also, ticket denominations have been rounded off to multiples of Rs.5 (7.2 US cents). As per the revised slab, a person traveling up to 15 km will have to pay Rs.5 (7.2 US cents), between 16 and 30 km Rs.10 (14 US cents), between 30 and 60 km Rs.15 (22 US cents) and between 60 and 100 km Rs.20 (29 US cents).

Train stations are ideal locations for small businesses to set up shops, because they are hubs of human interaction where hundreds or even thousands of people day and night come and go. Each person in this flow of foot traffic is a potential customer who might need a specific item or purchase on impulse while waiting for a train. To succeed with retail at a train station, one must provide an accessible and affordable shopping experience offering merchandise or services that travelers might not quickly find elsewhere enroute while travelling.

02. Business Problem

Train passengers as well as station and train employees need to eat breakfast, lunch, dinner and snacks. Although food sales are forbidden in some railway stations, many do offer merchants the opportunity to sell food. Foods that attract busy people on the go include egg sandwiches, fries, pizza, burgers, microwaveable or cold prepared meals. Beverages such as coffee, tea, wraps, bottled water, soda and juice also sell well. Thus, the main objective of the project will be to find ideal spots in the city where fast food retail chains can be put up, aiming at the above demographic, thereby helping the owners of the outlets to extract maximum profits out of them.

03. Data

The data for this project has been retrieved and processed through multiple sources, giving careful considerations to the accuracy of the methods used.
In following subparts I'll give the source of data accompanied with the code snippets used to extract the data from sources.

03.1 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 webpage - https://en.wikipedia.org/wiki/Category:Neighbourhoods_in_Kolkata .

Getting the source webpage and assigining the variable `source` to it and iniatilizing the `beautifulsoup` object to `soup`

[3]:
```
source = requests.get('https://en.wikipedia.org/wiki/Category:Neighbourhoods_in_Kolkata').text
soup = BeautifulSoup(source, 'lxml')
```

Initializing the `csv_writer` object and writing the name of the columns on it as the first row

[4]:
```
csv_file = open('kolkata.csv', 'w')
csv_writer = csv.writer(csv_file)
csv_writer.writerow(['Neighbourhood'])
```

[4]: 15

Scraping the page to extracting the list of neighbourhoods in Kolkata

[5]:
```
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):  # Gets all the neighbourhoods
    lists = mwcg [i].find_all('a')
    for list in lists:
        nbd = list.get('title') # Gets the title of the neighbourhood
        csv_writer.writerow([nbd]) # Writes the name of the neighbourhood in the csv file
csv_file.close()
```

03.2 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 intial dataframe. The URL used for API is https://maps.googleapis.com/maps/api/geocode/json?address={}&key={} where address is picked up from the data and my api key is used in key. Due to certain reasons the key is not mentioned anywhere in report or code.

Using the Google Maps Geocoding API

```
21]: import json

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_name, "AIzaS
    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
```

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. The credentials of FourSquare API are also removed from the project as well as report.

Using the FourSquare API on all neighbourhoods

```
explore_df_list = []

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

    try :
        ### Getting the data of neighbourhood
        nbd_name = df.loc[i, 'Neighbourhood']
        nbd_lat = df.loc[i, 'Latitude']
        nbd_lng = df.loc[i, 'Longitude']

        radius = 1000 # Setting the radius as 1000 metres
        LIMIT = 30 # Getting the top 30 venues

        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) # Flattens JSON

        # Filtering the columns
        filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.ln
        nearby = nearby.loc[:, filtered_columns]

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

        # Gets the categories
        nearby['Category'] = nearby.apply(get_category_type, axis=1)

        # Gets the data required
        for i, name in enumerate(nearby['Name']):
            s_list = nearby.loc[i, :].values.tolist()  # Converts the numpy array to a python list
            f_list = [nbd_name, nbd_lat, nbd_lng] + s_list
            explore_df_list.append(f_list)

    except Exception as e:
        pass
```

04. Methodology

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

4.1 Accuracy of the Geocoding API

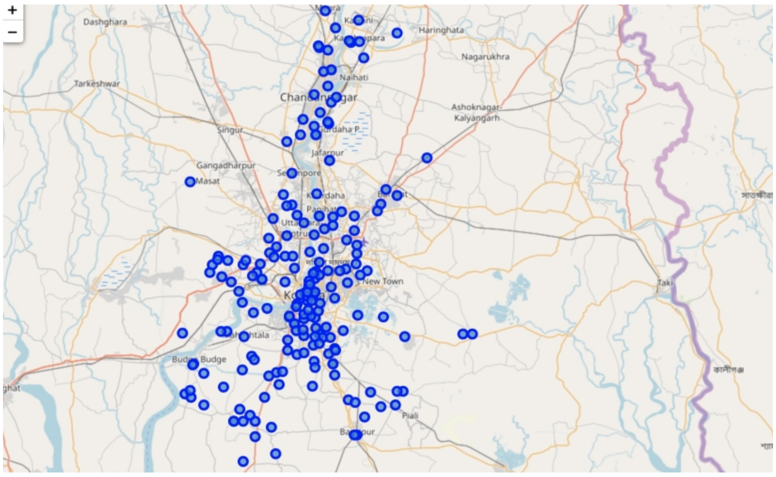
In the initial development phase with OpenCage Geocoder API, the number of erroneous results were of an appreciable amount, which led to the development of an algorithm to analyze the accuracy of the Geocoding API used. In the algorithm developed, Geocoding API from various providers were tested, and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

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 visualization 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)
```

Following image shows neighbourhoods of Kolkata generated by folium.



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()
```

After one hot encoding the data looks as follows:

	Neighbourhood	ATM	Afghan Restaurant	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Arcade	Art Gallery	...	Tea Room	Tex-Mex Restaurant	Thai Restaurant	Theme Restaurant
0	Agarpara	1	0	0	0	0	0	0	0	0	...	0	0	0	0
1	Agarpara	0	0	0	0	0	0	0	0	0	...	0	0	0	0
2	Agarpara	0	0	0	0	0	0	0	0	0	...	0	0	0	0
3	Ajoy Nagar	0	0	0	0	0	0	0	0	0	...	0	0	0	0
4	Ajoy Nagar	0	0	0	0	0	0	0	0	0	...	0	0	0	0

After aggregating values by neighbourhood :

	Neighbourhood	ATM	Afghan Restaurant	Airport	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Arcade	Art Gallery	...	Tea Room	Tex-Mex Restaurant	Thai Restaurant	Theme Restaurant
0	Agarpara	0.333333	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	Ajoy Nagar	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	Alipore	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	Alipur, Jaynagar	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	Amodghata	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 138 columns

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[‘Neighbourhooc

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)
```

It gives us following most common venues.

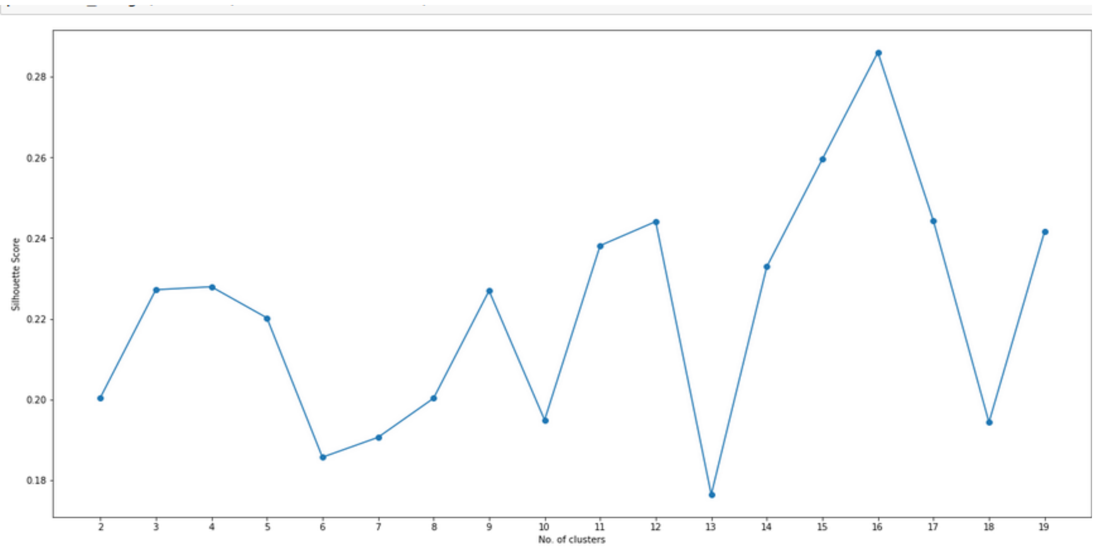
	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Mos Commor Venue
0	Agarpara	ATM	Train Station	Pharmacy	Airport	Flea Market	Cricket Ground	Department Store	Dessert Shop	Dhaba	Dine
1	Ajoy Nagar	Shopping Mall	Multiplex	Bakery	Grocery Store	Bus Station	Department Store	Diner	Dumpling Restaurant	Dhaba	Desser Shop
2	Alipore	Dessert Shop	South Indian Restaurant	Hotel	Chinese Restaurant	Pizza Place	Café	Italian Restaurant	Garden	Athletics & Sports	Women's Store
3	Alipur, Jaynagar	Dessert Shop	South Indian Restaurant	Hotel	Chinese Restaurant	Pizza Place	Café	Italian Restaurant	Garden	Athletics & Sports	Women's Store
4	Amodghata	Pharmacy	Electronics Store	Film Studio	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Eastern European Restaurant	Concert Ha

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

```
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
```

The corresponding plot is:



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.

K-Means clustering for the optimal number of clusters

```
[30]: kclusters = opt

      # Run k-means clustering
      kgc = kolkata_grouped_clustering
      kmeans = KMeans(n_clusters = kclusters, init = 'k-means++', random_state = 0).fit(kgc)

[31]: # Add clustering labels
      neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

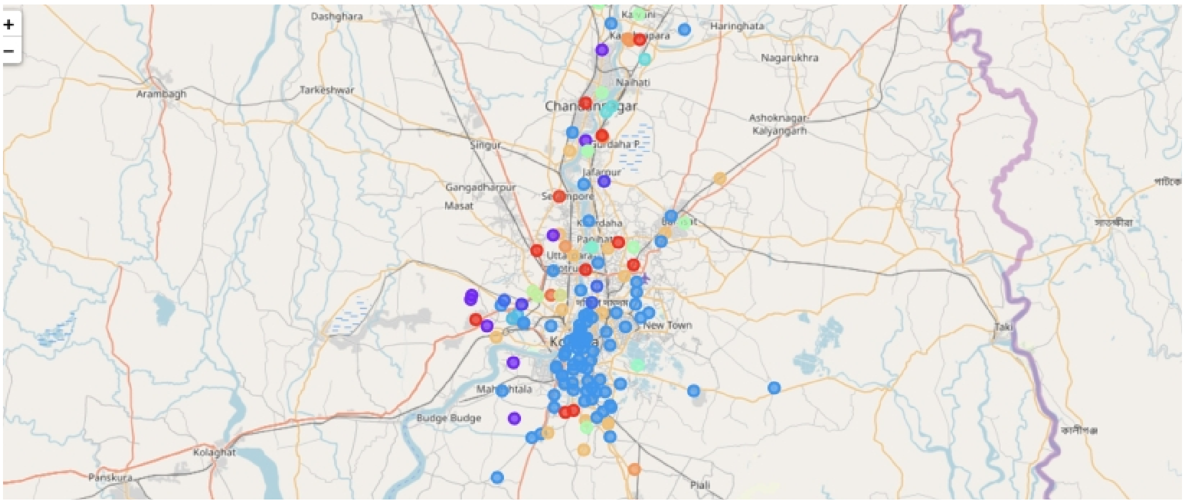
5. Results

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.

Visualizing the clusters

[33]:
```
# Create map
map_clusters = folium.Map(location=[kol_lat, kol_lng], zoom_start=11)

# Set color scheme for the clusters
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]

# Add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(kolkata_merged['Latitude'], kolkata_merged['Longitude'], kolkata_merged['Neigh
    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))

map_clusters
```

The following output is obtained (Clustered nieghbourhoods of Kolkata):



6. Discussion

After analyzing the various clusters produced by the Machine learning algorithm, cluster no.14, is a prime fit to solving the problem of finding a cluster with common venue as a train station mentioned before.

[56]:

	Neighbourhood	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
60	Bijpur, North 24 Parganas	Train Station	Women's Store	Electronics Store	Film Studio	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Eastern European Restaurant
123	Garshyamnagar	Train Station	Platform	Women's Store	Eastern European Restaurant	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Electronics Store
131	Halisahar	Train Station	Women's Store	Electronics Store	Film Studio	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Eastern European Restaurant
141	Hind Motor	Light Rail Station	Train Station	Women's Store	Electronics Store	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Eastern European Restaurant
186	Kodalia	Train Station	Women's Store	Electronics Store	Film Studio	Field	Fast Food Restaurant	Falafel Restaurant	Fabric Shop	Event Service	Eastern European Restaurant

As shown in image above the five places namely Bijpur, Garshyamnagar, Halisahar, Hind Motor and Kodalia fall in the outskirts of the city of Kolkata, hence the demographic of the population in these areas fall under the lower middle class of the society.

According to most organizations, like the World Bank and the Organization for the Economic Cooperation and Development (OECD), people living on less than US $2 a day are con- sidered poor. For those in the middle classes, the earnings typically lie in the range of US $10 to $100 per day, as expressed in the 2015 purchasing power parities.

India is expected to see a dramatic growth in the middle class, from 5 to 10 percent of the population in 2005 to 90 percent in 2039, by which time a billion people will be added to this group. In 2005, the mean per capita household expenditure was just US $3.20 per day, and very few households exceeded incomes of US $5 per day. Yet, by 2015, half the population had crossed this threshold. By 2025, half the Indian population is expected to surpass US $10 per day.

7. Conclusion

As the middle class will grow at a rapid rate in the next upcoming years, opening food outlets catered for that section of the society will see a massive increase in footfall, which would lead to a further increase in business.

If the food outlets have an average rate of US $0.5 equivalent to 15 percent of the per capita household expenditure, for their items, then profits can be expected to be high as the food rates are neither too low or too high for a person of the concerned demographic to spend. Assuming a footfall of 30 people getting off at these stations for each station, 100 trains passing through these stations, and a conversion rate of 20 percent, ordering only one meal, a daily turnover of around US $300 can be expected from these outlets per station.

8. External Links

Notebook: https://github.com/Me-Quanta/Coursera_Capstone/blob/master/Coursera%20Capstone%20Project%20.ipynb

PPT: https://github.com/Me-Quanta/Coursera_Capstone/blob/master/Final%20Presentation.pdf