# Coursera Capstone Project : Applied Data Science

**Raghunath Siripudi**

## Business Problem

Mumbai is a busy city. People here are always in a hurry. Local train passengers as well as best bus users and employees need to eat breakfast, lunch, dinner. Although food sales are forbidden in some railway stations, many do offer merchants the opportunityto sell food. Foods that attract people in a hurry are easy snacks and lite food like sandwiches, fries, pizza, burger. 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 exact locations idea in the city for food retail chains, aiming at the above demographic, thereby helping the owners of the outlets to extract maximum profits out of them.

## Data Collection

There are several sources to collect data for above problem. We have choosen WIKIPEDIA page of wikipedia.org/wiki/List_of_neighbourhoods_in_Mumbai for neighbourhood data of mumbai man and in later part we collect the venue data through foursquare api and model the data and prdeict the locations for ideal spots to keep a food chain in the Mumbai city. At first the data of the neighbourhoods in Mumbai can be extracted out by web scraping using BeautifulSoup library for Python. The neighbourhood data is taken from a Wikipedia webpage mentioned above. The file contents from Mumbai csv is retrieved into a Pandas DataFrame. The latitude and longitude of the neighbourhoods are retrieved using geospacial data.The location values are then stored into the intial dataframe.From the location data obtained after the above data frame , the data is pointed 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. By choosing a methodology we obtain required locations

```
Code:
# Get the text data od neighbours in mumbai from wikipedia
source = requests.get('https://en.wikipedia.org/wiki/List_of_neighbourh
oods_in_Mumbai').text
 soup = BeautifulSoup(source, 'lxml')print(soup.find('li'))

table_post = soup.find('li')mumareas = table_post.find_all('span')neigh
bourhood = []
```

```
for i in range(0, len(mumareas),2):
    neighbourhood.append(mumareas[i+1].text.strip())
 df_mumbai= pd.DataFrame(data=[neighbourhood]).transpose()
df_mumbai.columns = ['Neighbourhood']

#to find each area coordinates
from geopy.geocoders import Nominatim
latitudes = [] # Initializing the latitude array
 longitudes = [] # Initializing the longitude array
 for area in df_mumbai["Neighbourhood"] :
place_name = area + ",Mumbai, India"
geolocator = Nominatim(user_agent="googleapis")
 location = geolocator.geocode(area)
 latitudes.append(location.latitude)
longitudes.append(location.longitude)
df_mumbai['Latitude']=latitudes
df_mumbai['Longitude'] = longitudes
```

**Methodology**
A thorough analysis of the principles of methods, rules, and postulates employed g=have been made in order to ensure the inferences to be made are as accurate as possible.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 our analysis.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 clear map.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.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.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.

Code:
```
explore_df_list = []
for i, nbd_name in enumerate(df_mumbai['Neighbourhood']):

    try :
        ### Getting the data of neighbourhood
        nbd_name = df_mumbai.loc[i, 'Neighbourhood']
        nbd_lat = df_mumbai.loc[i, 'Latitude']
        nbd_lng = df_mumbai.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.l
ocation.lat', 'venue.location.lng']
        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

df_venue = pd.DataFrame([item for item in explore_df_list])
df_venue.columns = ['Neighbourhood', 'Neighbourhood Latitude', 'N
eighbourhood Longitude', 'Venue Name', 'Venue Category', 'Venue 
Latitude', 'Venue Longitude']
df_venue.head()
```

```
mumbai_onehot = pd.get_dummies(df_venue[['Venue Category']], pref
ix="", prefix_sep="")
# Add neighborhood column back to dataframe
mumbai_onehot['Neighbourhood'] = df_venue['Neighbourhood']
# Move neighborhood column to the first column
fixed_columns = [mumbai_onehot.columns[-1]] + mumbai_onehot.col
umns[:-1].values.tolist()
mumbai_onehot = mumbai_onehot[fixed_columns]
mumbai_grouped = mumbai_onehot.groupby('Neighbourhood').mean().
reset_index()
num_top_venues = 10indicators = ['st', 'nd', 'rd']
# Create columns according to number of top venuescolumns = ['N
eighbourhood']for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, i
ndicators[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'] = mumbai_grouped['
Neighbourhood']
for ind in np.arange(mumbai_grouped.shape[0]):
    neighbourhoods_venues_sorted.iloc[ind, 1:] = return_most_comm
on_venues(mumbai_grouped.iloc[ind, :], num_top_venues)
neighbourhoods_venues_sorted.head()
from sklearn.cluster import KMeans# set number of clusters
kclusters = 10
mumbai_grouped_clustering = mumbai_grouped.drop(columns=['Neigh
bourhood'])
```

```
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(mumbai_
grouped_clustering)
# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]

neighbourhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels
_)
```

```
mumbai_merged = df_mumbai
```

```
mumbai_merged = mumbai_merged.join(neighbourhoods_venues_sort
ed.set_index('Neighbourhood'), on='Neighbourhood')
```

```
mumbai_merged.head()
```

**Analysis**

The neighbourhoods are divided into n clusters where n is the nu mber of clusters found using the optimal approach. The clustered neighbourhoods are visualized using different colours so as to mak e them distiguishable.After through analysis of 10 clusters we hav e a clear view that cluste of jogeswari has least common fast foo d restaurants so we can suggest a fast food restaurant there. And also we can further analysis the rush areas and suggest them.