IBM Data Science Capstone Project (C10-W5)

Opening a new Fast-Food Restaurant in Los Angeles, California

Final Report

By: Phaneendra Annamaraju

1 Contents

2 Introduction

Opening a new fast-food restaurant in Los Angeles

Los Angeles is very density populated city in west coast of USA. It is one of the major cultural hubs in the world with immigrants from many countries live in harmony. It is well known for Hollywood movie industry, night life and restaurants. Los Angeles has a very big market for restaurants and fast-food restaurants are very famous due to the tourism. Due to this, there are thousands of restaurants in the city and there is huge competition in the business. It is a very challenging task for any restaurant chain to plan their business expansion.

In this project we will try to find an optimal location for a restaurant. Specifically, this report will be targeted to stakeholders interested in opening a **fast-food restaurant** in **Los Angeles, USA**.

2.1.1 Problem Statement

A fast-food restaurant chain wants to expand their business and wants to know suitable neighborhood to open a new restaurant in Los Angeles.

Since there are lots of restaurants in Los Angeles we need to detect **locations that are not already crowded with restaurants** yet well populated

2.1.2 Objective

The objective is to use data science methodology and machine learning algorithms to answer the following critical questions in order recommend potential location for a new fast-food restaurant in Los Angeles

- Which areas have potential Fast Food Restaurants Market?
- Which areas are lacking Fast Food Restaurants?
- What are the suitable neighborhoods in Los Angeles for opening a new Fast-Food Restaurant?

2.1.3 Target Audience

This is very useful for some of the major fast food restaurant chains such as Burger King, Subway, McDonald's etc… It will help them in preparing a strategy for their business expansion also understand the areas with potential market for fast food restaurants.

It will provide much more value add with further analysis on other influencing factors, measures and key external indicators. For example: impact of other similar restaurants, user ratings, market share, per capita income etc…

3 Approach

To answer the above questions,

- Analyze number of restaurants in each neighborhood
- Population density of each neighborhood
- Select the neighborhood that has lowest restaurant to population density ratio

We will use Machine learning algorithms to perform the above analysis

- k-Mean clustering will provide groups of neighborhoods with similar categories of restaurants
    - This makes it easy to do analysis on the data
- Exploratory analysis on the clusters and the venues
    - We will review number of restaurants, nearby fast food restaurants and population density

4 Data

4.1 Data requirement

- We first need list of all neighborhoods in Los Angeles
- Population density of each neighborhood
- Latitude and longitude coordinates of each neighborhood
- Venue data, specifically, list of restaurants around each neighborhood

4.2 Data collection

- LA Times for neighborhood details and population density
    - http://maps.latimes.com/neighborhoods/neighborhood/list/
    - http://maps.latimes.com/neighborhoods/population/density/neighborhood/list/
- Geocoder library in python for latitude and longitude coordinates of neighborhoods
- Foursquare API for restaurants around each neighborhood

5 Methodology

In this project, we will focus on detecting neighborhoods in Los Angeles have low restaurants to population density.

Below are the steps we will follow

5.1 Data Collection

5.1.1 Web scraping – get list of neighborhoods

(Webscraping) Get a list of neighbood in Los Angeles from LA times website

From: http://maps.latimes.com/neighborhoods/neighborhood/list/

[5]:
```
url = 'http://maps.latimes.com/neighborhoods/neighborhood/list/'
html_data = requests.get(url).text
soup = BeautifulSoup(html_data, 'html.parser')
tables = soup.find_all('table')
table_index = 0
n_count = len(tables[table_index].tbody.findAll('tr'))
print("Number of neighborhoods codes: ", n_count)
```

```
Number of neighborhoods codes:  272
```

[7]:
```
# Review the data
print(df_postal_codes_with_ll.shape)
df_postal_codes_with_ll.head(10)
```

```
(265, 5)
```

[7]:

	Neighborhood	Region	Address	Latitude	Longitude
0	Acton	Antelope Valley	Acton, California, United States	34.480742	-118.186838
1	Adams-Normandie	South L.A.	Adams-Normandie, Los Angeles, California, Unit...	34.031788	-118.300247
2	Agoura Hills	Santa Monica Mountains	Agoura Hills, California, 91301, United States	34.147910	-118.765704
3	Agua Dulce	Northwest County	Agua Dulce, California, United States	34.496382	-118.325635
4	Alhambra	San Gabriel Valley	Alhambra, California, United States	34.093042	-118.127060
5	Alondra Park	South Bay	Alondra Park, California, 90506, United States	33.890134	-118.335139
6	Altadena	Verdugos	Altadena, California, 91001, United States	34.186316	-118.135233
7	Angeles Crest	Angeles Forest	Los Angeles Air Force Base (Pacific Crest Hous...	33.720966	-118.311107
8	Arcadia	San Gabriel Valley	Arcadia, California, United States	34.136207	-118.040150
9	Arleta	San Fernando Valley	Arleta, Los Angeles, California, United States	34.241327	-118.432205

5.1.2 Get population density

(Webscraping) Get population density for each neighborhood from LA Times website

From: http://maps.latimes.com/neighborhoods/population/density/neighborhood/list/

[8]:
```
url = 'http://maps.latimes.com/neighborhoods/population/density/neighborhood/list/'
html_data = requests.get(url).text
soup = BeautifulSoup(html_data, 'html.parser')
tables = soup.find_all('table')
table_index = 0
for i, t in enumerate(tables):
    if("Population per Sqmi" in str(t)):
        table_index = i

n_count = len(tables[table_index].tbody.findAll('tr'))
print("Table index: ",table_index)
print("Number of neighborhoods: ", n_count)

table_contents=[]
i = 0
for row in tables[table_index].tbody.findAll('tr'):
    c = row.findAll('td')
    cell = {}
    if c == None:
        pass
    else:
        cell['Neighborhood'] = c[1].a.text.strip()
        cell['Population per Sqmi'] = c[2].text.replace(',','').strip()
        table_contents.append(cell)
```

```
df_population_density = pd.DataFrame(table_contents)
df_population_density.head(10)
```

```
Table index:  1
Number of neighborhoods:  265
```

[8]:

	Neighborhood	Population per Sqmi
0	Koreatown	42611
1	Westlake	38214
2	East Hollywood	31095
3	Pico-Union	25352
4	Maywood	23638
5	Harvard Heights	23473
6	Hollywood	22193
7	Walnut Park	22028
8	Palms	21870
9	Adams-Normandie	21848

5.1.3 Geocoder library in python

Get longitude and latidue of each neighborhood using Geopy library

Note: this code timeout after processing around 500 records so I ran few at a time and saved the data into a file for frequent use

```
# Get the neighborhoods and their geo locations... this takes time and sometimes fail. so I did it porttion of data at a time and saved it to
geolocator = Nominatim(user_agent="geoapiExercises1")

table_contents=[]
i = 0
for row in tables[table_index].tbody.findAll('tr'):
    c = row.findAll('a')
    cell = {}
    if c == None:
        pass
    else:
        cell['Neighborhood'] = c[0].text
        cell['Region'] = c[1].text
        geo_ll = geolocator.geocode(c[0].text + ', CA, USA')
        if geo_ll:
            cell['Address'] = geo_ll.address
            cell['Latitude'] = geo_ll.latitude
            cell['Longitude'] = geo_ll.longitude
            table_contents.append(cell)
        else:
        print('unable to find geo location for ' + c[0].text)
    if i in [20, 40, 60, 80, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300]:
        print(i)
    i = i + 1

df_postal_codes_with_ll = pd.DataFrame(table_contents)
```

```
# Review the data
print(df_postal_codes_with_ll.shape)
df_postal_codes_with_ll.head(10)
```

```
(265, 5)
```

	Neighborhood	Region	Address	Latitude	Longitude
0	Acton	Antelope Valley	Acton, California, United States	34.480742	-118.186838
1	Adams-Normandie	South L.A.	Adams-Normandie, Los Angeles, California, Unit...	34.031788	-118.300247
2	Agoura Hills	Santa Monica Mountains	Agoura Hills, California, 91301, United States	34.147910	-118.765704
3	Agua Dulce	Northwest County	Agua Dulce, California, United States	34.496382	-118.325635
4	Alhambra	San Gabriel Valley	Alhambra, California, United States	34.093042	-118.127060
5	Alondra Park	South Bay	Alondra Park, California, 90506, United States	33.890134	-118.335139
6	Altadena	Verdugos	Altadena, California, 91001, United States	34.186316	-118.135233
7	Angeles Crest	Angeles Forest	Los Angeles Air Force Base (Pacific Crest Hous...	33.720966	-118.311107
8	Arcadia	San Gabriel Valley	Arcadia, California, United States	34.136207	-118.040150
9	Arleta	San Fernando Valley	Arleta, Los Angeles, California, United States	34.241327	-118.432205

5.1.4 Foursquare API

- Get list of restaurants around each neighborhood
- We should also identify and tag fast-food restaurants
- And capture number restaurants as well as number of fast-food restaurants into dataframes

```
column_names=['Region', 'Neighborhood', 'Neighborhood Latitude', 'Neighborhood Longitude', 'Venue', 'Venue Latitude', 'Venue Longitude', 'Venue Category', 'Venue ID']
restaurants=pd.DataFrame(columns=column_names)
neighborhoods = pd.DataFrame(columns=['Region', 'Neighborhood', 'Latitude', 'Longitude', 'Population per Sqmi', 'Restaurant Count', 'Restaurant to Population Ratio'])

count = 1

# LOOP THROUGH EACH NEIGHBORHOOD AND GET VENUES
for row in df_la.loc[:,['Region', 'Neighborhood', 'Latitude', 'Longitude', 'Population per Sqmi']].values.tolist():
    Region, Neighborhood, Latitude, Longitude, Population = row
    venue_details=[]

    # GET VENUES FROM FOURSQUARE
    url = 'https://api.foursquare.com/v2/venues/explore?&categoryId={}&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
        CATEGORY_ID,
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        Latitude,
        Longitude,
        RADIUS,
        LIMIT)
    venue_data = requests.get(url).json()["response"]['groups'][0]['items']

    # INSERT VENUES INTO A DATA FRAME
    # return only relevant information for each nearby venue
    try:
        venue_details.append([(
            Region,
            Neighborhood,
            Latitude,
            Longitude,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name'],
            v['venue']['categories'][0]['id']) for v in venue_data])
    except KeyError:
            pass

    if len(venue_details[0]) > 0:
        venues = pd.DataFrame([item for venue_list in venue_details for item in venue_list])
        venues.columns = column_names
        restaurants = restaurants.append(venues, ignore_index=True)
        neighborhoods = neighborhoods.append({
                                              'Region' : Region,
                                              'Neighborhood' : Neighborhood,
                                              'Latitude' : Latitude,
                                              'Longitude' : Longitude,
                                              'Population per Sqmi' : int(Population),
                                              'Restaurant Count' : len(venues),
                                              'Fast Food Restaurant Count': len(venues[venues['Venue Category'] == 'Fast Food Restaurant']),
                                              'Restaurant to Population Ratio' : (len(venue_details) / int(Population))
                                             }, ignore_index=True)
```

5.2Pre-process data

5.2.1 Combine all data from sources by merging the data frames

- We will come up with below are the 2 key data frames for our analysis
  - neighborhoods : holds a unique list of neighborhoods and details of each (such as coordinates, number of restaurants, number of fast-food restaurants etc...)
  - restaurants : holds list of restaurants in each neighborhood and their coordinates along with the restaurant category

```
neighborhoods.drop('Restaurant to Population Ratio', axis=1, inplace=True)
print("Neighborhoods data frame shape: ", neighborhoods.shape)
print("Restaurants data frame shape: ", restaurants.shape)
neighborhoods.head()
```

```
Neighborhoods data frame shape:  (226, 7)
Restaurants data frame shape:  (3303, 9)
```

	Region	Neighborhood	Latitude	Longitude	Population per Sqmi	Restaurant Count	Fast Food Restaurant Count
0	South L.A.	Adams-Normandie	34.031788	-118.300247	21848	7	0.0
1	Santa Monica Mountains	Agoura Hills	34.147910	-118.765704	2495	6	0.0
2	Northwest County	Agua Dulce	34.496382	-118.325635	99	6	0.0
3	San Gabriel Valley	Alhambra	34.093042	-118.127060	11275	36	0.0
4	Verdugos	Altadena	34.186316	-118.135233	4900	11	1.0

5.2.2 On-hot encoding with restaurant categories

One hot encoding

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


# add neighborhood column back to dataframe
neighborhood_onehot['Neighborhood'] = restaurants['Neighborhood']

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

# Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category
neighborhood_grouped = neighborhood_onehot.groupby('Neighborhood').mean().reset_index()
neighborhood_grouped
```

	Neighborhood	African Restaurant	American Restaurant	Andhra Restaurant	Argentinian Restaurant	Asian Restaurant	Australian Restaurant	BBQ Joint	Bagel Shop	Bakery	...	Taco Place	Taiwanese Restaurant	Tapas Restaurant	Tex-Mex Restaurant	Thai Restaurant	Theme Restaurant	Ukrainian Restaurant
0	Adams-Normandie	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.000000	...	0.142857	0.0	0.0	0.0	0.000000	0.0	0.0
1	Agoura Hills	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
2	Agua Dulce	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.166667	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
3	Alhambra	0.0	0.027778	0.0	0.0	0.055556	0.0	0.027778	0.000000	0.055556	...	0.000000	0.0	0.0	0.0	0.027778	0.0	0.0
4	Altadena	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.090909	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
221	Willowbrook	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
222	Wilmington	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
223	Windsor Square	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.105263	0.105263	...	0.052632	0.0	0.0	0.0	0.000000	0.0	0.0
224	Winnetka	0.0	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0
225	Woodland Hills	0.0	0.000000	0.0	0.0	0.032258	0.0	0.032258	0.064516	0.000000	...	0.000000	0.0	0.0	0.0	0.032258	0.0	0.0

226 rows × 109 columns

5.2.3 Identify 10 most common restaurant categories

Now create data frame of neighborhoods with 10 most common Venues

```
# create data frame with the new data

## First, let's write a function to sort the venues in descending order.
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]

## Now let's create the new dataframe and display the top 10 venues for each neighborhood.

num_top_venues = 10
indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
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['Neighborhood'] = neighborhood_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head()
```

	Neighborhood	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
0	Adams-Normandie	Sushi Restaurant	Food	Latin American Restaurant	Taco Place	African Restaurant	Mongolian Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Persian Restaurant
1	Agoura Hills	Indian Restaurant	Pizza Place	Deli / Bodega	Chinese Restaurant	Breakfast Spot	Snack Place	African Restaurant	Moroccan Restaurant	Polish Restaurant	Poke Place
2	Agua Dulce	Pizza Place	Bakery	Café	Mexican Restaurant	Restaurant	African Restaurant	Moroccan Restaurant	Polish Restaurant	Poke Place	Peruvian Restaurant
3	Alhambra	Burger Joint	Chinese Restaurant	Seafood Restaurant	Vietnamese Restaurant	Asian Restaurant	Bakery	Diner	Sandwich Place	Food	Poke Place
4	Altadena	Food Truck	Pizza Place	Diner	Sandwich Place	Fast Food Restaurant	Bakery	Halal Restaurant	Food	Burger Joint	African Restaurant

5.3Clustering

- Perform clustering on the neighborhoods based on the restaurant categories

```
grouped_clustering = neighborhood_grouped.drop('Neighborhood', 1)

# set number of clusters
kclusters = 8

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

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

```
array([6, 6, 6, 6, 6, 6, 7, 3, 6, 6], dtype=int32)
```

Add the cluster labels to the neighborhoods_venues_sorted dataframe.
And Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

# merge north_york_grouped with north_york_data to add latitude/longitude for each neighborhood
neighbhorhoods_merged = neighborhoods.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

neighbhorhoods_merged.head() # check the last columns!
```

	Region	Neighborhood	Latitude	Longitude	Population per Sqmi	Restaurant Count	Fast Food Restaurant Count	Cluster Labels	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
0	South L.A.	Adams-Normandie	34.031788	-118.300247	21848	7	0.0	6	Sushi Restaurant	Food	Latin American Restaurant	Taco Place	African Restaurant	Mongolian Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Persian Restaurant
1	Santa Monica Mountains	Agoura Hills	34.147910	-118.765704	2495	6	0.0	6	Indian Restaurant	Pizza Place	Deli / Bodega	Chinese Restaurant	Breakfast Spot	Snack Place	African Restaurant	Moroccan Restaurant	Polish Restaurant	Poke Place
2	Northwest County	Agua Dulce	34.496382	-118.325635	99	6	0.0	6	Pizza Place	Bakery	Café	Mexican Restaurant	Restaurant	African Restaurant	Moroccan Restaurant	Polish Restaurant	Poke Place	Peruvian Restaurant
3	San Gabriel Valley	Alhambra	34.093042	-118.127060	11275	36	0.0	6	Burger Joint	Chinese Restaurant	Seafood Restaurant	Vietnamese Restaurant	Asian Restaurant	Bakery	Diner	Sandwich Place	Food	Poke Place
4	Verdugos	Altadena	34.186316	-118.135233	4900	11	1.0	6	Food Truck	Pizza Place	Diner	Sandwich Place	Fast Food Restaurant	Bakery	Halal Restaurant	Food	Burger Joint	African Restaurant

- Combine clusters for further analysis

Combine clusters and common venues into one list and pivot.This is needed for next plot.

```
column_headers=['Cluster Labels', 'Venue Category']
c_list = pd.DataFrame(columns=column_headers)

for i in neighbhorhoods_merged['Cluster Labels'].unique():
    for v in ['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 Venu
        c = neighbhorhoods_merged[neighbhorhoods_merged['Cluster Labels']==i].loc[:,['Cluster Labels', v]]
        c.columns = column_headers
        c_list = c_list.append(c)

c_list.head()
```

	Cluster Labels	Venue Category
0	6	Sushi Restaurant
1	6	Indian Restaurant
2	6	Pizza Place
3	6	Burger Joint
4	6	Food Truck

5.4Folium map

- Visualize data in a map using folium mapping library in python
- We will show a map of all neighbourhoods in all clusters (using **k-means clustering**)

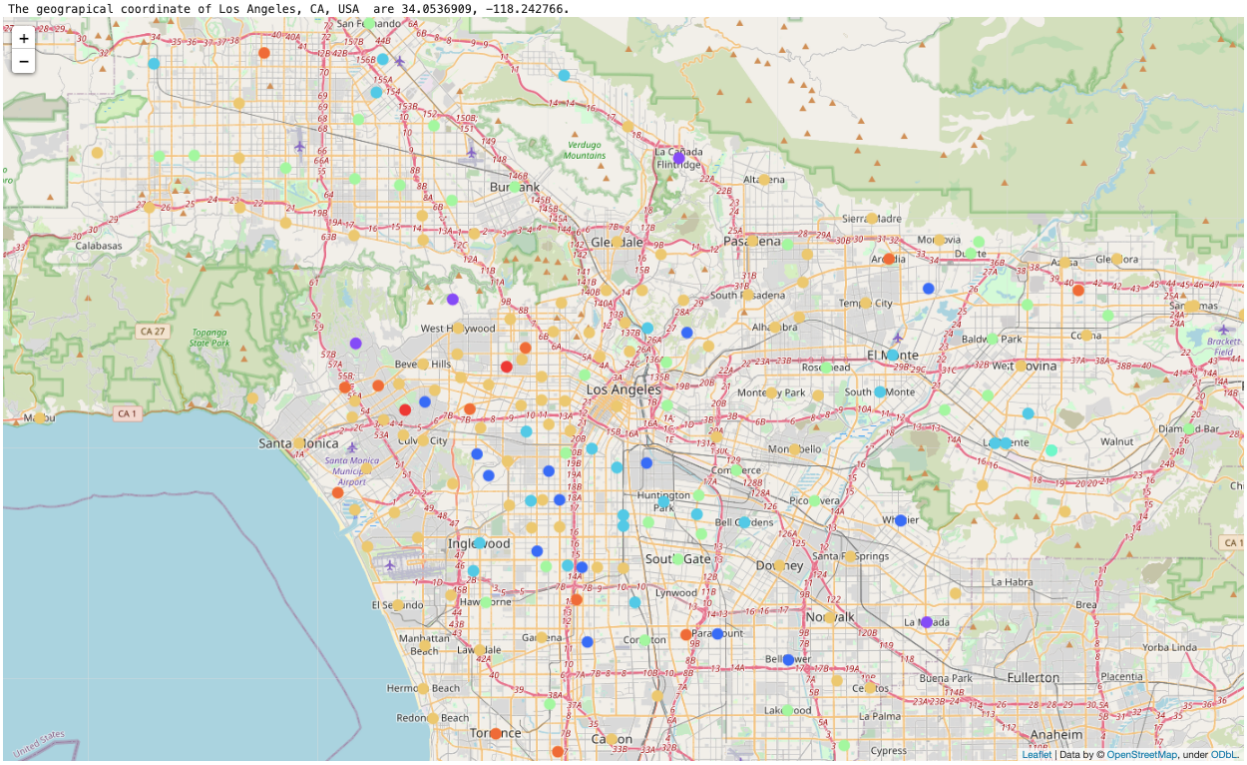
```
# create map
address = 'Los Angeles, CA, USA'
geolocator = Nominatim(user_agent="to_explorer4")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of', address, ' are {}, {}.'.format(latitude, longitude))
map_clusters = folium.Map(location=[latitude, longitude], 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 = [plt_colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(neighbhorhoods_merged['Latitude'], neighbhorhoods_merged['Longitude'], neighbhorhoods_merged['Neighborhood'], neighbhorhoods_merged['Cluster Lab
    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=1).add_to(map_clusters)

map_clusters
```

```
The geograpical coordinate of Los Angeles, CA, USA  are 34.0536909, -118.242766.
```



5.5Exploratory Analysis and Observations

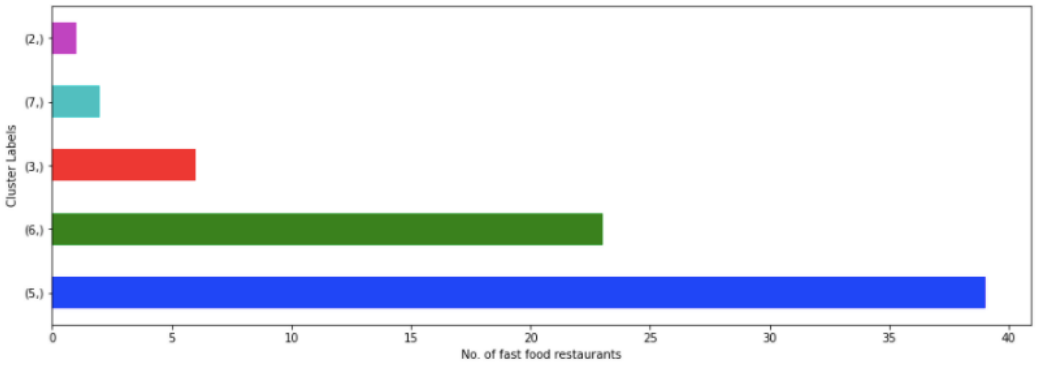
5.5.1 Number of fast-food restaurants in each cluster

- Observation from below graph: ¶
  - Cluster 5 has highest number of fast food restaurants
  - Cluster 2 has lowest
  - We can also notice that clusters 0, 1 and 4 do not have any fast food restaurants

```
# Fast food restaurants in each cluster
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'g', 'b']

c_list[c_list['Venue Category'] == 'Fast Food Restaurant'][['Cluster Labels']].value_counts().plot.barh(color=colors, figsize=(15,5))
#plt.xticks(np.arange(0, 15, 5))
plt.xlabel('No. of fast food restaurants')
```

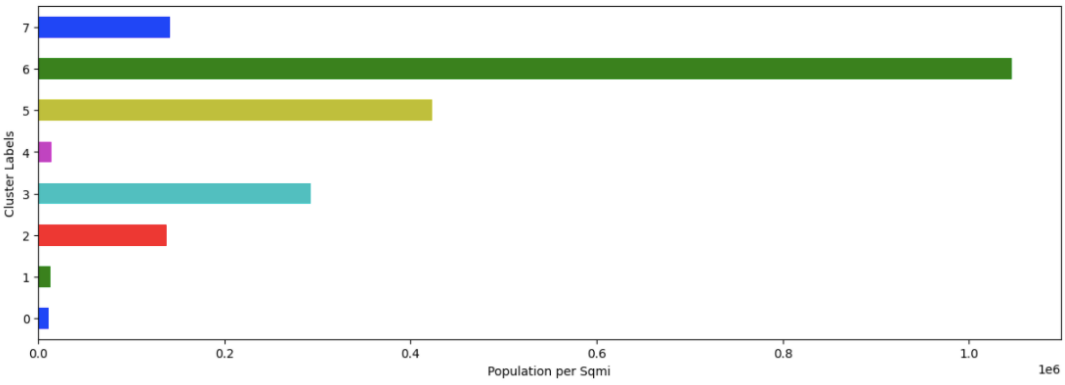
```
Text(0.5, 0, 'No. of fast food restaurants')
```



5.5.2 Population density in each cluster

- Observation from below graph:
  - Population density is very less in Clusters 0, 1 and 4 so these cannot be suitable places to open restaurants.
  - Hence we need to further review the clusters 2, 3, 6, 5, & 7

```
plt.figure(figsize=(9,5), dpi = 100)
plt.xlabel('Population per Sqmi')
neighbhorhoods_merged.sort_values(['Population per Sqmi'],ascending=False).groupby('Cluster Labels')['Population per Sqmi'].sum().head(20).plot.barh(color=colors, figsize=(15,5))
plt.show()
```



5.5.3 Restaurants to population ratio

Calculate restaurants to population ratio for each cluster and identify

```
x = pd.DataFrame(neighbhorhoods_merged.sort_values(['Population per Sqmi'],ascending=False).groupby('Cluster Labels')['Population per Sqmi'].sum())
y = pd.DataFrame(neighbhorhoods_merged.sort_values(['Restaurant Count'],ascending=False).groupby('Cluster Labels')['Restaurant Count'].sum())
x = x.reset_index()
y = y.reset_index()
s_data = x.merge(y, on='Cluster Labels')
s_data['Restaurant to Population Ratio'] = s_data['Restaurant Count'] / s_data['Population per Sqmi']
s_data.sort_values(['Restaurant to Population Ratio', 'Population per Sqmi'])
```

	Cluster Labels	Population per Sqmi	Restaurant Count	Restaurant to Population Ratio
4	4	14186	2	0.000141
0	0	10979	2	0.000182
2	2	137891	43	0.000312
3	3	293266	183	0.000624
7	7	142022	96	0.000676
1	1	12713	16	0.001259
5	5	423520	557	0.001315
6	6	1046242	2404	0.002298

5.5.4 Scatter plot - Population density and restaurant count

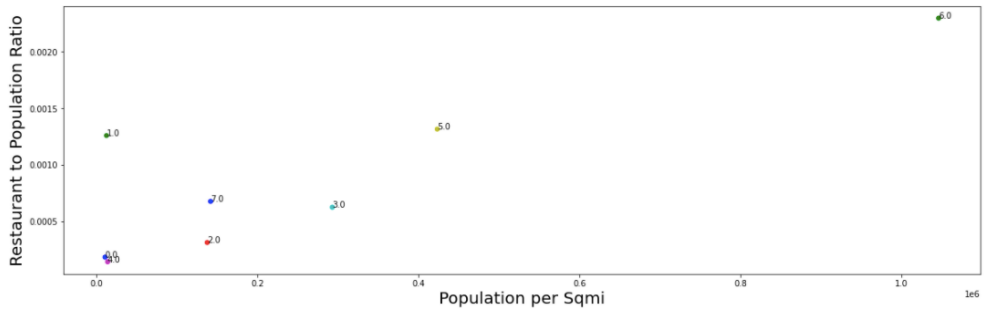
- Observation from below scatter plot: ¶
    - Clusters 2 and 3 have very less number restaurants given the population density.
    - Where cluster 5 and 6 already have good number of restaurants.

```
fig = plt.figure(1,figsize=(20,6))
plt.subplot(111)
plt.scatter(
    s_data['Population per Sqmi'],
    s_data['Restaurant to Population Ratio'],
    marker='o', color=colors, s=25)

for i, row in s_data.iterrows():
    plt.annotate(row["Cluster Labels"], xy=(row["Population per Sqmi"],row["Restaurant to Population Ratio"]))

plt.xlabel('Population per Sqmi',size=20)
plt.ylabel('Restaurant to Population Ratio',size=20)

plt.show()
```



5.5.5 Most suitable neighborhood

Sort the data in selected clusters and find the neighbourhood that is most suitable for opening a new fast-food restaurant which has

- more population
- less number of restaurants (including all categories)
- and less number of fast-food restaurants
- Observation from below table:
    - With in Clusters 2 and 3, 'Lennox' neighborhood has least number of restaurants, given the population density.

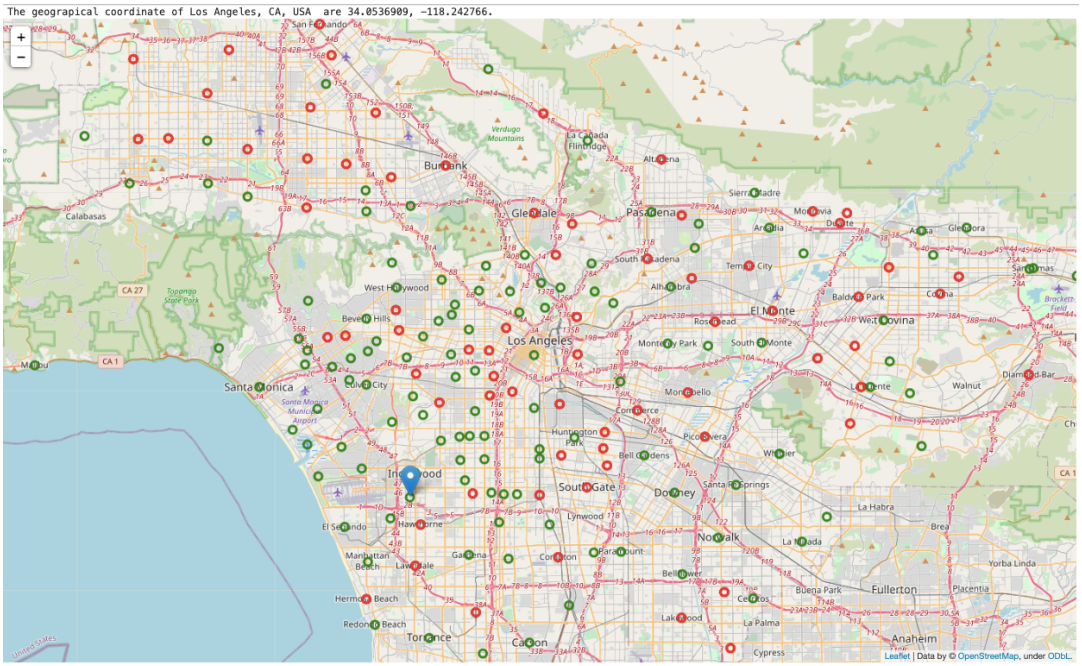
- Also, there are no fast-food restaurants near by.
- Hence, it is the most suitable place to open a new fast food restaurant

```
# Look at cluster 2 & 3
neighbhorhoods_merged[(neighbhorhoods_merged['Cluster Labels'] == 2) | (neighbhorhoods_merged['Cluster Labels'] == 3)].sort_values(['Population per Sqmi', 'Restaurant Count'],asc
```

	Region	Neighborhood	Latitude	Longitude	Population per Sqmi	Restaurant Count	Fast Food Restaurant Count	Cluster Labels	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	10
112	South Bay	Lennox	33.939031	-118.357443	21557	9	0.0	3	Mexican Restaurant	Taco Place	Pizza Place	Seafood Restaurant	Food Truck	Middle Eastern Restaurant	Poke Place	Peruvian Restaurant	Persian Restaurant	Re
92	Southeast	Huntington Park	33.982704	-118.212034	20223	4	0.0	3	Fried Chicken Joint	Mexican Restaurant	Burger Joint	Food	African Restaurant	Moroccan Restaurant	Polish Restaurant	Poke Place	Pizza Place	Re
89	South L.A.	Historic South-Central	34.016230	-118.267308	19474	9	1.0	3	Mexican Restaurant	Restaurant	Taco Place	Fast Food Restaurant	Donut Shop	Seafood Restaurant	American Restaurant	Tex-Mex Restaurant	Tapas Restaurant	Re
33	South L.A.	Central-Alameda	34.004015	-118.247784	18760	4	0.0	3	Donut Shop	Taco Place	Food	Mexican Restaurant	African Restaurant	Mongolian Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Re
198	South L.A.	Vermont-Slauson	33.983691	-118.291542	18577	5	0.0	2	Food	Sandwich Place	Burger Joint	African Restaurant	Mongolian Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Persian Restaurant	Re
199	South L.A.	Vermont Square	34.001945	-118.300213	17798	4	0.0	2	Food	Food Truck	Burger Joint	Mongolian Restaurant	Polish Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Persian Restaurant	Re
20	Southeast	Bell Gardens	33.969456	-118.150395	17762	8	0.0	3	Mexican Restaurant	Donut Shop	Latin American Restaurant	Fried Chicken Joint	Burger Joint	New American Restaurant	African Restaurant	Moroccan Restaurant	Polish Restaurant	
70	South L.A.	Florence-Firestone	33.967426	-118.243307	16805	5	0.0	3	Mexican Restaurant	Bakery	Food	African Restaurant	Mongolian Restaurant	Polish Restaurant	Poke Place	Pizza Place	Peruvian Restaurant	Re
97	South L.A.	Jefferson Park	34.027234	-118.317576	16300	3	0.0	3	Mexican	Fried Chicken	Burger Joint	Middle Eastern	Polish	Poke	Pizza	Peruvian	Persian	

5.5.6 Review the results in map

- Observation from below map:
  - Red circles has fast food restaurants in the neighborhoods
  - where as, Green circles do not have any fast food restaurants but has other categories of restaurants
  - The selected neighborhood i.e. 'Lennox' is highlighted and you can see there are limited restaurants in the surrounding neighborhood and no fast-food restaurants near by.
  - 'Lennox' has 9 restaurants in total however zero fast food restaurants. Hence, it is the most suitable place to open a new fast food restaurant.



6 Observations

- Cluster 5 has most number of fast food restaurants, where as, cluster 2 has lowest.
- We can also notice that clusters 0, 1 and 4 do not have any fast food restaurants. However, population in these clusters is very less so these cannot be good places to open restaurants.
- Where in cluster 5 and 6 already have good number of restaurants so these are the potential good markets for fast food restaurants
- Clusters 2 and 3 have very less number restaurants given the population density so opening new restaurant near by these clusters will be a good option
- Within Clusters 2 and 3, 'Lennox' neighbourhood has least number of restaurants, given the population density. And there are no Fast food restaurants nearby. Therefore, it is the most suitable place to open a new fast food restaurant

7 Results

We now answered all the questions from the problem statement

1. Which areas have potential Fast Food Restaurant Market?
    - Clusters 5 and 6 have the large number of restaurants
2. Which areas are lacking Fast Food Restaurants?
    - Clusters 2 and 3 have very a smaller number of fast-food restaurants
3. What are the suitable neighborhoods in Los Angeles for opening a new Fast-Food Restaurant?
    - 'Lennox' from Cluster 3 is the most suitable place for a new fast-food restaurant

8 Conclusion

- We were able to identify the suitable neighborhood for opening a new fast-food restaurant by using k-Mean clustering algorithm and exploratory analysis on the data.
    - As you can see in the map in 5f, combining exploratory analysis with clustered data provided better insight and helped with decision making
- We can improve our modeling by including more influencing factors, measures, and key external indicators and identifying the correlation

    For example:

    - impact of other similar restaurants
    - user ratings
    - market share
    - demographic data
    - per capita income etc...