Additional data analysis for choosing the location of a new Chinese restaurant in Los Angeles, Ca.

Arshak Mkhitaryan
Data Scientist

January 2020

The business problem

In no more than 48 hours the CEO of a Chinese restaurant business is going to pitch a project to the board of stakeholders. The agenda of the project is related to expanding the business by opening another restaurant at a new location in Los Angeles.

But how to choose a location for a new venue?

She approaches me as the data scientist of the company with a request for additional insight and visualization on the matter. Specifically, she is interested in the mapping and clustering of potential regions within the city.

The final problem can be summarised as follows: cluster regions in Los Angeles based on information regarding Chinese restaurants and present relevant insight and visualizations that can be incorporated into the final presentation for the board of executives in. The results should be presented to the CEO in 24 hours so that she has an adequate amount of time to incorporate them into her presentation.

Background

How does one find the best location for a venue?

The problem of finding the best location for a new venue might seem trivial at a glance but it’s not. There are many criteria to consider to support the decision. Some of them are as follows:

1. The number of similar restaurants in the neighborhood.
2. The house prices in the neighborhood.
3. The average income levels of the residents.
4. The average percentage of people eating out in the area.
5. The trends and migration patterns in the area that might be taken as a proxy for the demand in a particular cuisine.
6. The number of business centers in the area.
7. The history of restaurants that closed or prospered in the past and the reasons behind their failure or success.
8. etc.

And the list goes on and on. As we can see, it is obvious that a comprehensive prediction of success of a venue based on the location and subsequently an effective choice of such locations requires a large amount of data and complex data analysis as well as a deep understanding of underlying business processes.

That is why it is very important to define the limits of the given problem and possible solutions right away.

Nonetheless, it’s worth undertaking quick research providing additional insights into the best practices and the state of the art. During my initial research, I found two main approaches frequently applied to the problem in question as well as combinations of those.

The first approach heavily relies on domain knowledge and business expertise. The ability of the business team to “sense” best locations relying on their experience.

The second approach that is becoming more and more popular with these sorts of problems is using Geolocational data of individuals together with locational data of venues to extrapolate information such as clustering customers into meaningful groups by their preferences and then matching those clusters with clusters of venues.

It’s worth mentioning that both approaches are very similar to each other in the fact that they require an enormous amount of information. In one case that information is gained by humans through years of hands-on experience of opening new locations and running businesses. In the other, it’s the amount of data collected in time and extrapolated using unsupervised machine learning techniques such as the Travel Time Factorization Model (TTFM)[1].

What makes it so hard?

My current understanding is that the problem is complicated not only by the amount of information needed to choose the location but also because of the Dynamically changing nature of the data.

Let’s, for example, try to answer these types of questions. How our venue will be affected by the economic changes in the region? How would it be affected by another restaurant closing or opening nearby? How would it be affected by migration patterns within the area, such as the number of people that might be interested in visiting our new Chinese restaurant?

Having all of the complexity of the problem and the limitations of time and resources we first must define the scope and depth of our analysis.

In this particular case, the business decision would be mostly based on the expertise of the business team and sparingly supplemented by our analysis and visualization.

[1] For more information check out this study https://arxiv.org/abs/1801.07826 or this blogpost https://blog.safegraph.com/opening-a-new-restaurant-ditch-the-guesswork-and-turn-to-machine-learning-5276f44dd408.

Interest

Business owners, as well as upcoming entrepreneurs, will find the reports helpful in deciding on the best location for a Chinese restaurant in Los Angeles. Although this analysis is not providing the final answer it is a good informational supplement.

Data

The data used in the analysis are as follows:

1. The list of notable districts and neighborhoods of the city of Los Angeles, California.
2. Source:
   https://en.wikipedia.org/wiki/List_of_districts_and_neighborhoods_of_Los_Angeles

3. This list will be scraped and cleaned in Python using the Beautiful Soup library and then stored in a Pandas Dataframe.

4. The latitude and longitude coordinates of each neighborhood will be obtained using Geocoder and then stored in a Pandas DataFrame.

   Location data from Foursquare.
   Source: https://foursquare.com/
   Data will be acquired through a developer's account using Places API and then stored in a Pandas Dataframe.

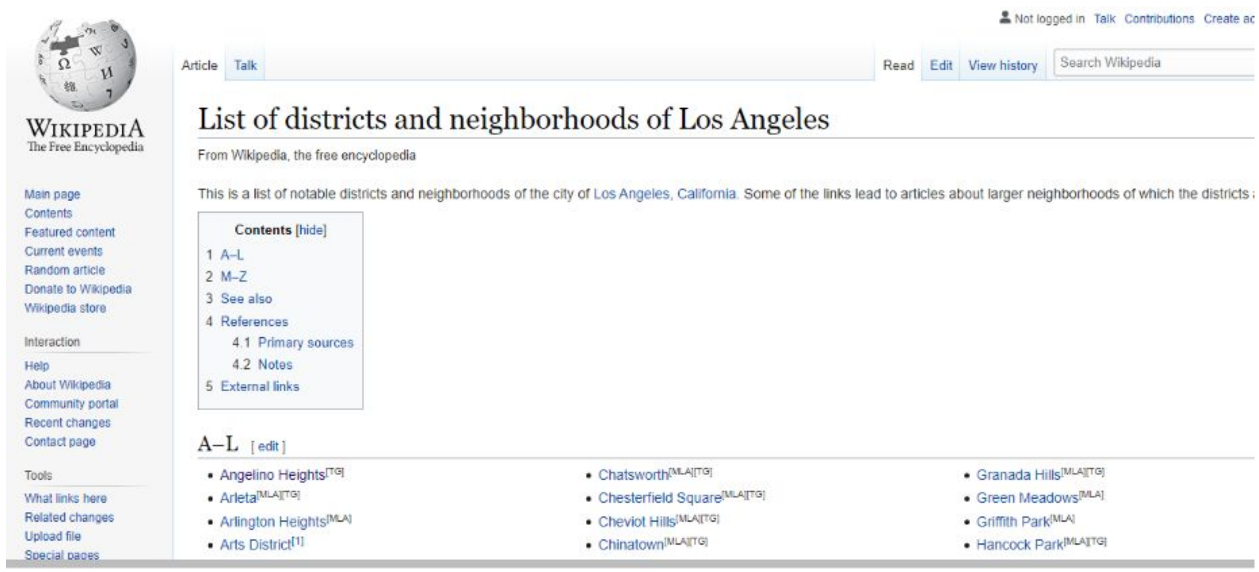
All of the Data above will be merged, processed and used to fit the K-means clustering machine learning algorithm.

The clusters then will be visualized using the Folium library.

Methodology

Data acquisition and cleaning.

To obtain the data I started by scraping a list of LA neighborhoods from a wikipedia page https://en.wikipedia.org/wiki/List_of_districts_and_neighborhoods_of_Los_Angeles.



List of districts and neighborhoods of Los Angeles

From Wikipedia, the free encyclopedia

This is a list of notable districts and neighborhoods of the city of Los Angeles, California. Some of the links lead to articles about larger neighborhoods of which the districts

Contents [hide]

1 A–L
2 M–Z
3 See also
4 References
4.1 Primary sources
4.2 Notes
5 External links

A–L [edit]

- Angelino Heights[TG]
- Arleta[MLA][TG]
- Arlington Heights[MLA]
- Arts District[1]
- Chatsworth[MLA][TG]
- Chesterfield Square[MLA][TG]
- Cheviot Hills[MLA][TG]
- Chinatown[MLA][TG]
- Granada Hills[MLA][TG]
- Green Meadows[MLA]
- Griffith Park[MLA]
- Hancock Park[MLA][TG]

For that I used Beautiful Soup library as shown below.

```
#importing the libraries
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

```
#defining the url
url ="https://en.wikipedia.org/wiki/List_of_districts_and_neighborhoods_of_Los_Angeles"
```

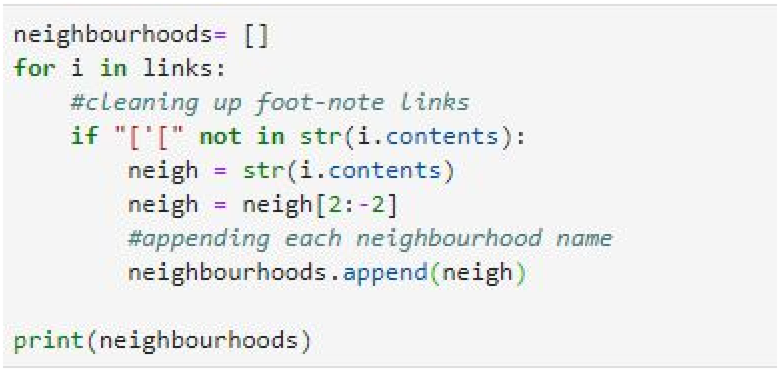
```
#get the raw data from the url
data = requests.get(url)
```

```
#geting html from the previous data
soup = BeautifulSoup(data.text, 'html.parser')
```

```
#selecting the links usig html tags and classes
links = soup.select('div.div-col ul li a')
links[:5]
```

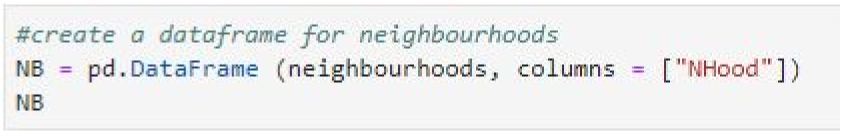
```
[<a href="/wiki/Angelino_Heights,_Los_Angeles" title="Angelino Heights, Los Ange
 <a href="/wiki/Arleta,_Los_Angeles" title="Arleta, Los Angeles">Arleta</a>,
 <a href="/wiki/Arlington_Heights,_Los_Angeles" title="Arlington Heights, Los An
 <a href="/wiki/Arts_District,_Los_Angeles" title="Arts District, Los Angeles">A
 <a href="#cite_note-VisitorsMap-1">[1]</a>]
```

Obtained data obviously needed to be cleaned.

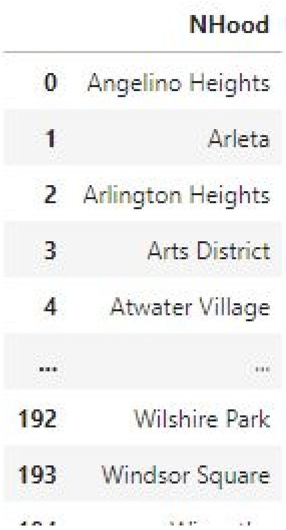


```
neighbourhoods= []
for i in links:
    #cleaning up foot-note links
    if "['[" not in str(i.contents):
        neigh = str(i.contents)
        neigh = neigh[2:-2]
        #appending each neighbourhood name
        neighbourhoods.append(neigh)

print(neighbourhoods)
```

Then the data was put into Pandas DataFrame.

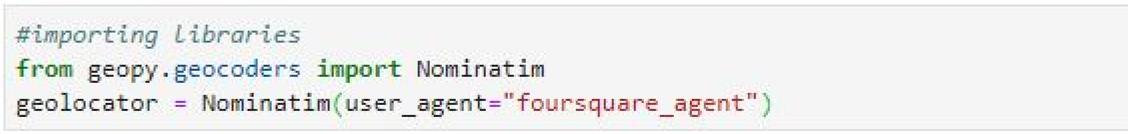


```
#create a dataframe for neighbourhoods
NB = pd.DataFrame (neighbourhoods, columns = ["NHood"])
NB
```

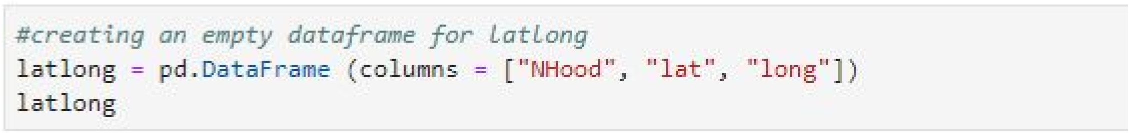


	NHood
0	Angelino Heights
1	Arleta
2	Arlington Heights
3	Arts District
4	Atwater Village
...	...
192	Wilshire Park
193	Windsor Square

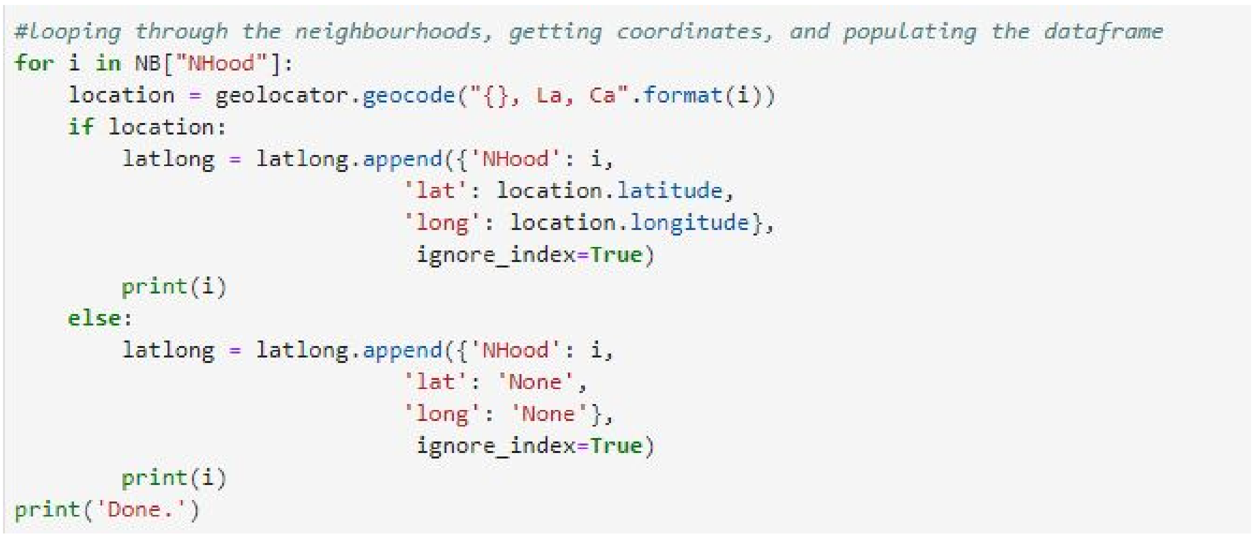
Importing the Geocoder libraries. Creating a Pandas dataframe and populating it with Geocoder data.



```
#importing libraries
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="foursquare_agent")
```

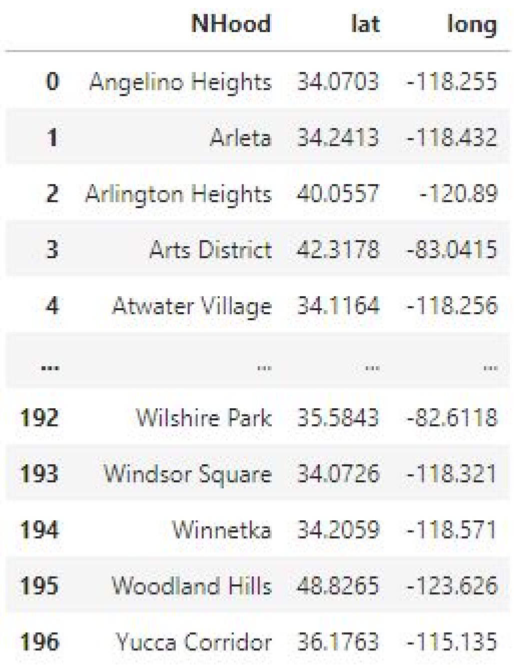


```
#creating an empty dataframe for latlong
latlong = pd.DataFrame (columns = ["NHood", "lat", "long"])
latlong
```



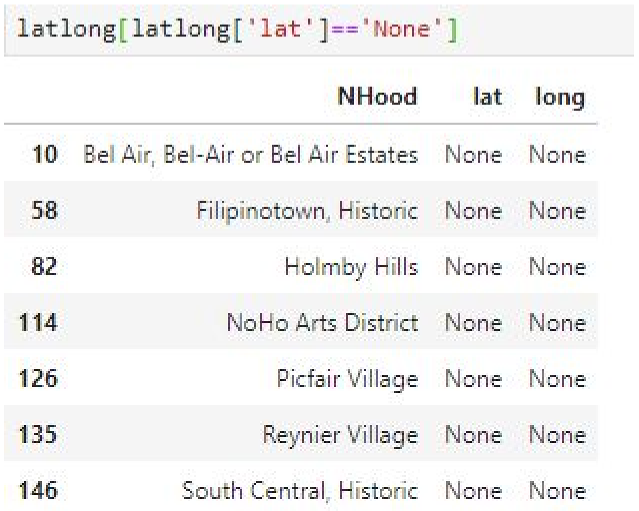
```
#looping through the neighbourhoods, getting coordinates, and populating the dataframe
for i in NB["NHood"]:
    location = geolocator.geocode("{}, La, Ca".format(i))
    if location:
        latlong = latlong.append({'NHood': i,
                                 'lat': location.latitude,
                                 'long': location.longitude},
                                  ignore_index=True)
        print(i)
    else:
        latlong = latlong.append({'NHood': i,
                                 'lat': 'None',
                                 'long': 'None'},
                                  ignore_index=True)
        print(i)
print('Done.')
```

This is the final data frame including LA neighborhoods with respective latitude and longitude coordinates.



	NHood	lat	long
0	Angelino Heights	34.0703	-118.255
1	Arleta	34.2413	-118.432
2	Arlington Heights	40.0557	-120.89
3	Arts District	42.3178	-83.0415
4	Atwater Village	34.1164	-118.256
...	...	...	...
192	Wilshire Park	35.5843	-82.6118
193	Windsor Square	34.0726	-118.321
194	Winnetka	34.2059	-118.571
195	Woodland Hills	48.8265	-123.626
196	Yucca Corridor	36.1763	-115.135

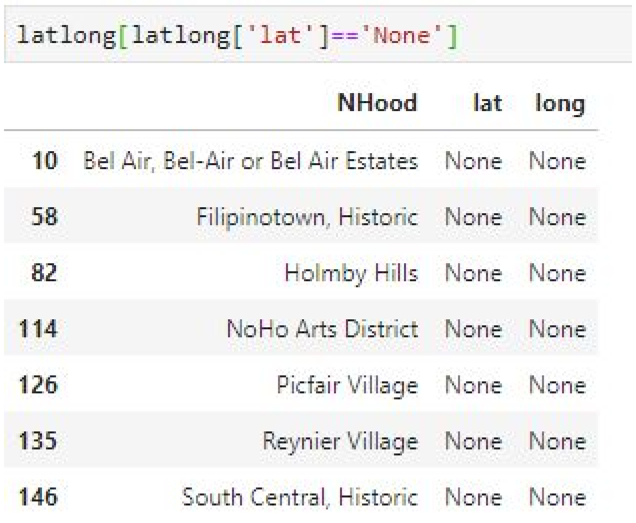
Noticing first problems with Geocoder data.



```
latlong[latlong['lat']=='None']
```

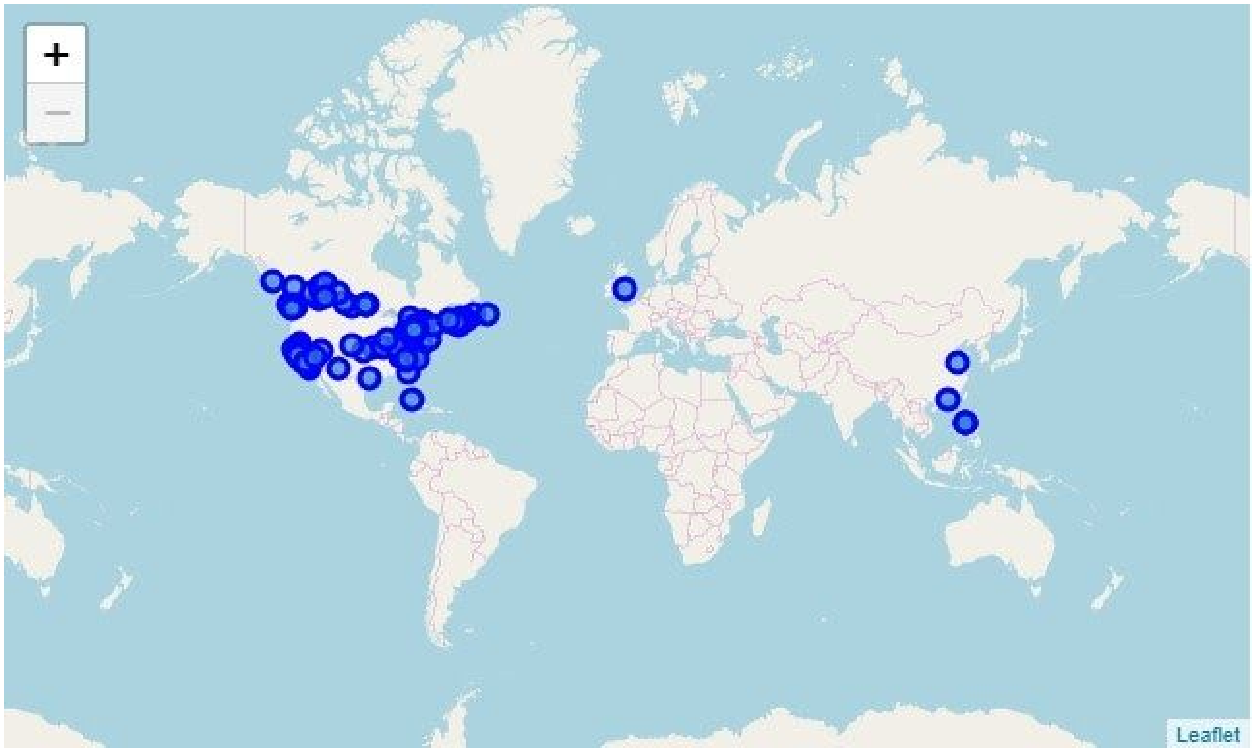
	NHood	lat	long
10	Bel Air, Bel-Air or Bel Air Estates	None	None
58	Filipinotown, Historic	None	None
82	Holmby Hills	None	None
114	NoHo Arts District	None	None
126	Picfair Village	None	None
135	Reynier Village	None	None
146	South Central, Historic	None	None

As we see there are some missing values. After adding the values manually based on google search I proceeded to visualizing the neighborhood using Folium.



```python
latlong[latlong['lat']=='None']
```

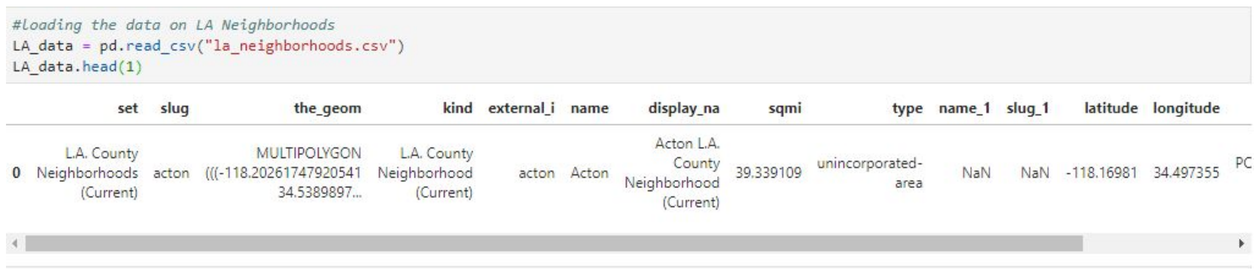
	NHood	lat	long
10	Bel Air, Bel-Air or Bel Air Estates	None	None
58	Filipinotown, Historic	None	None
82	Holmby Hills	None	None
114	NoHo Arts District	None	None
126	Picfair Village	None	None
135	Reynier Village	None	None
146	South Central, Historic	None	None



After a few attempts and unsatisfying results I decided to not lose any more time (especially since the time of the analysis was limited) and find LA neighborhood data online.

The data had been found at
https://usc.data.socrata.com/api/views/9utn-waje/rows.csv?accessType=DOWNLOAD

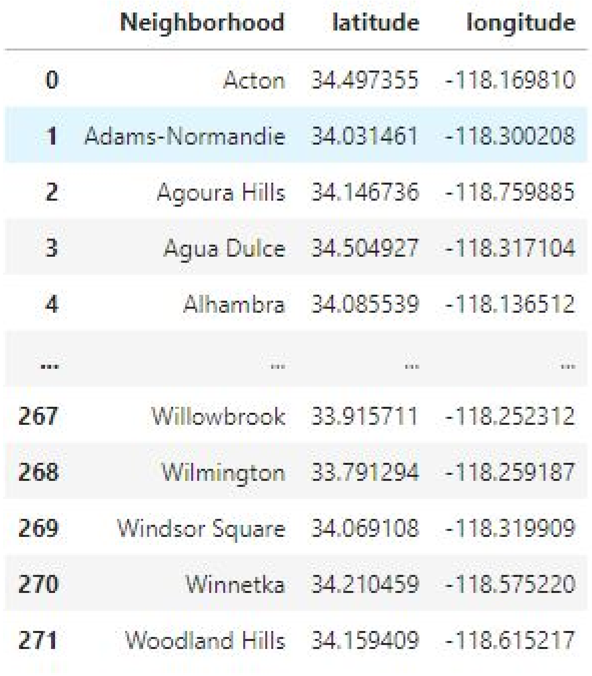
I started a new notebook and loaded the data.



```python
#loading the data on LA Neighborhoods
LA_data = pd.read_csv("la_neighborhoods.csv")
LA_data.head(1)
```

	set	slug	the_geom	kind	external_i	name	display_na	sqmi	type	name_1	slug_1	latitude	longitude	
0	L.A. County Neighborhoods (Current)	acton	MULTIPOLYGON (((-118.20261747920541 34.5389897...	L.A. County Neighborhood (Current)	acton	Acton	Acton L.A. County Neighborhood (Current)	39.339109	unincorporated-area	NaN	NaN	-118.16981	34.497355	PC

After selecting and renaming necessary columns (latitudes and longitudes were mixed up, which can be seen in the picture above) the final dataset had been obtained.

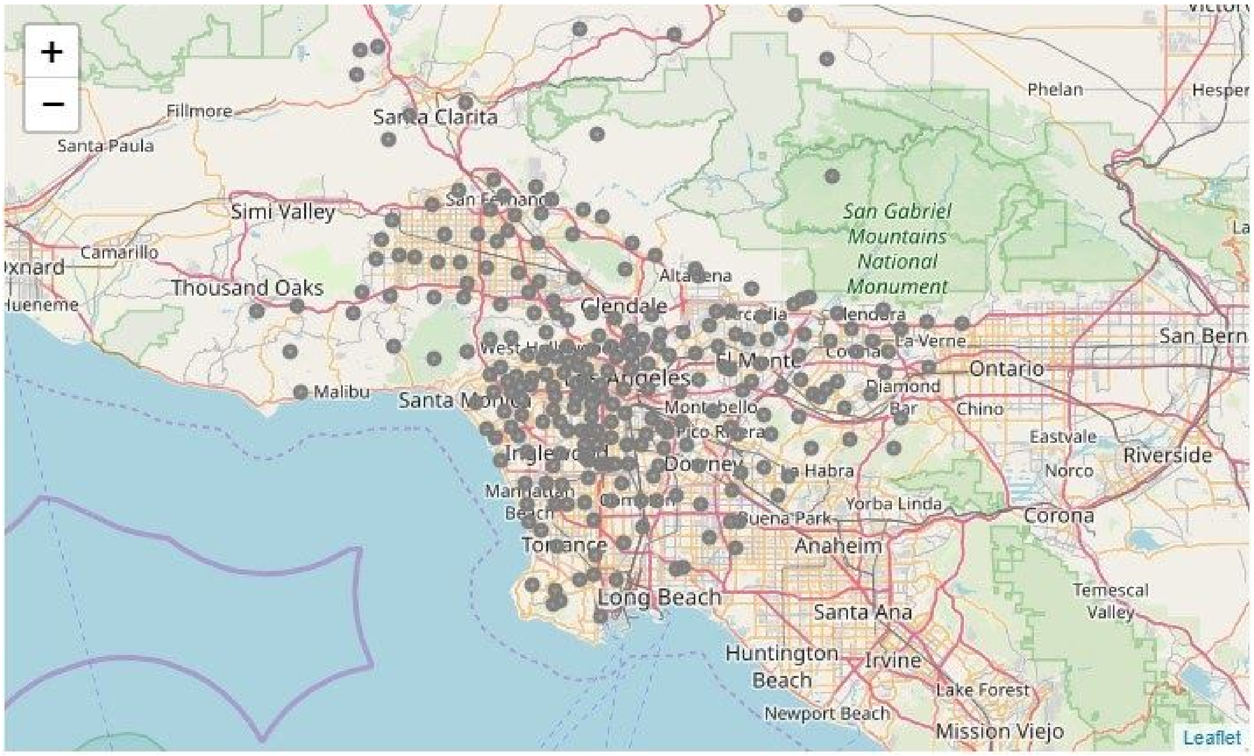


	Neighborhood	latitude	longitude
0	Acton	34.497355	-118.169810
1	Adams-Normandie	34.031461	-118.300208
2	Agoura Hills	34.146736	-118.759885
3	Agua Dulce	34.504927	-118.317104
4	Alhambra	34.085539	-118.136512
...	...	...	...
267	Willowbrook	33.915711	-118.252312
268	Wilmington	33.791294	-118.259187
269	Windsor Square	34.069108	-118.319909
270	Winnetka	34.210459	-118.575220
271	Woodland Hills	34.159409	-118.615217

Finally!)

Fast forward I used Folium to visualise all the neighborhoods in LA

```
#create the map
# create map of LA using latitude and longitude values
map_LA = folium.Map(location=[LA[0], LA[1]], zoom_start=9)

# add markers to map
for lat, lng, label in zip(LA_data['latitude'], LA_data['longitude'], LA_data['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='grey',
        fill=True,
        fill_color='grey',
        fill_opacity=0.7,
        parse_html=False).add_to(map_LA)

map_LA
```

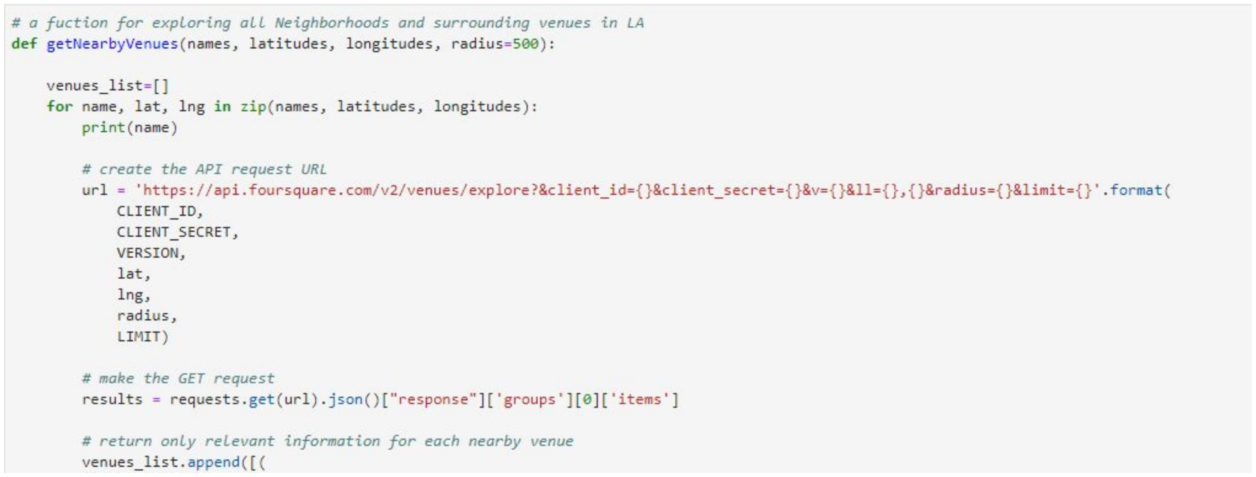


Obtaining venue information on neighborhoods from Foursquare.

```
#Define Foursquare Credentials and Version
CLIENT_ID = '5UHQPSVP1JJB3J35A4FLLBW4J512AVWBOSEIWUUKLR0KV0BZ' # your Foursquare ID
CLIENT_SECRET = 'A2EQBFHX2X44RL0QJQMUOUU4YH4SOTB12ZMTZZAQVYPF2YG5' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

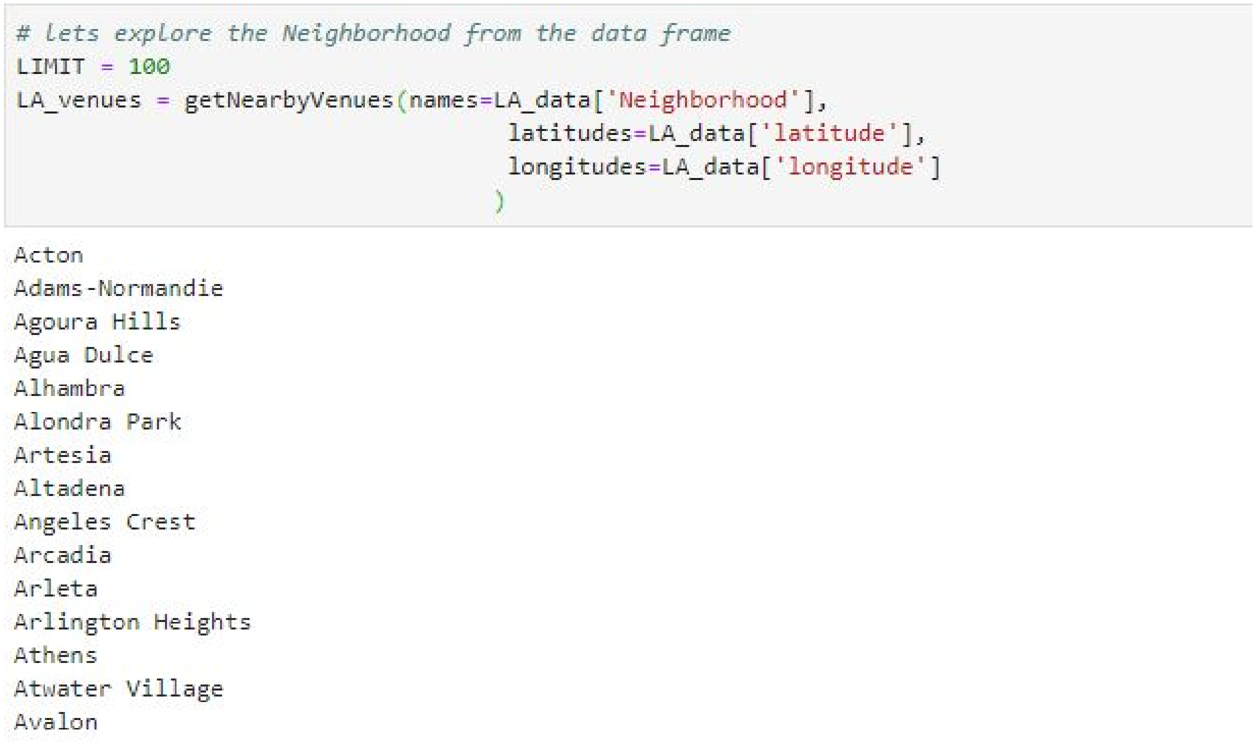
```
Your credentails:
CLIENT_ID: 5UHQPSVP1JJB3J35A4FLLBW4J512AVWBOSEIWUUKLR0KV0BZ
CLIENT_SECRET:A2EQBFHX2X44RL0QJQMUOUU4YH4SOTB12ZMTZZAQVYPF2YG5
```

I created a function for retrieving all of the respective venues based on neighborhoods.



```python
# a fuction for exploring all Neighborhoods and surrounding venues in LA
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
```

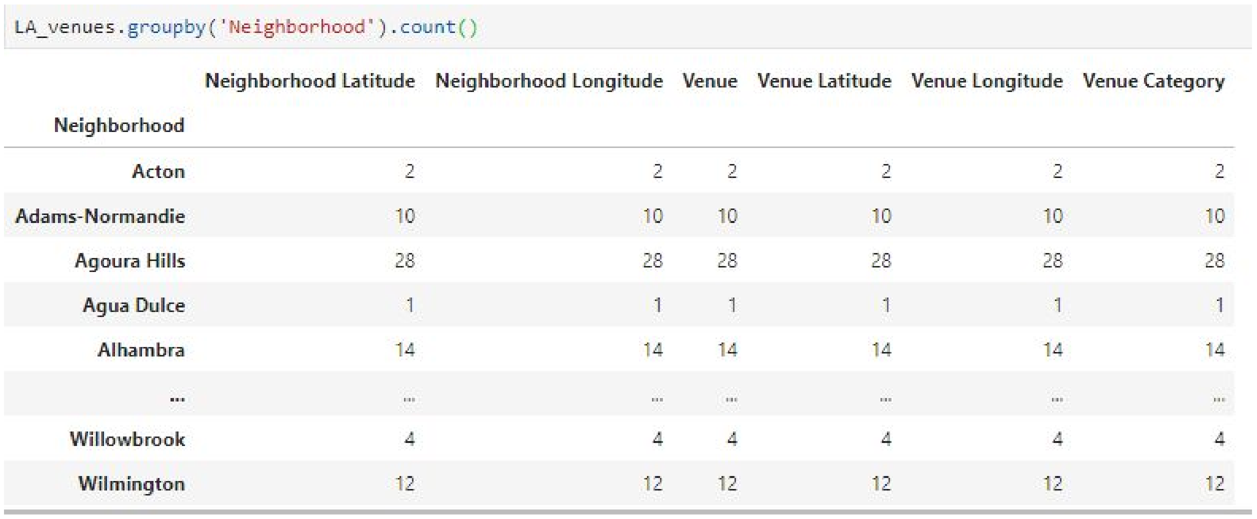
This is just a part of the function, you can see the full version in the notebook.



```python
# lets explore the Neighborhood from the data frame
LIMIT = 100
LA_venues = getNearbyVenues(names=LA_data['Neighborhood'],
                            latitudes=LA_data['latitude'],
                            longitudes=LA_data['longitude']
                           )
```

```
Acton
Adams-Normandie
Agoura Hills
Agua Dulce
Alhambra
Alondra Park
Artesia
Altadena
Angeles Crest
Arcadia
Arleta
Arlington Heights
Athens
Atwater Village
Avalon
```

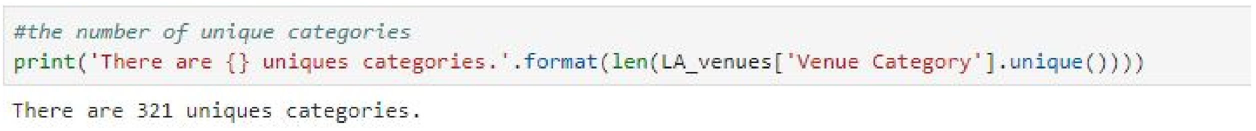
I then continued exploratory data analysis by checking the number of venues returned for each venue.



```
LA_venues.groupby('Neighborhood').count()
```

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Acton	2	2	2	2	2	2
Adams-Normandie	10	10	10	10	10	10
Agoura Hills	28	28	28	28	28	28
Agua Dulce	1	1	1	1	1	1
Alhambra	14	14	14	14	14	14
...	...	...	...	...	...	...
Willowbrook	4	4	4	4	4	4
Wilmington	12	12	12	12	12	12

And the number of unique categories.



```
#the number of unique categories
print('There are {} uniques categories.'.format(len(LA_venues['Venue Category'].unique())))
```

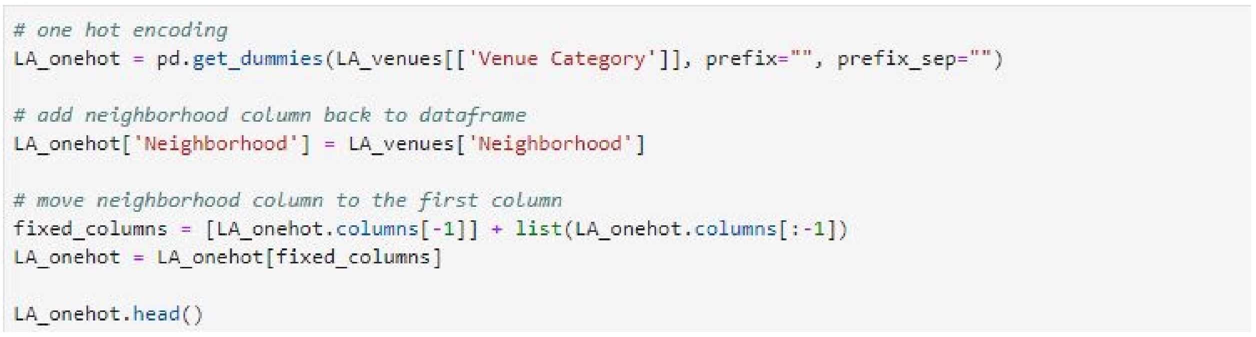
```
There are 321 uniques categories.
```

Which is 321.

Preparing the data for clustering

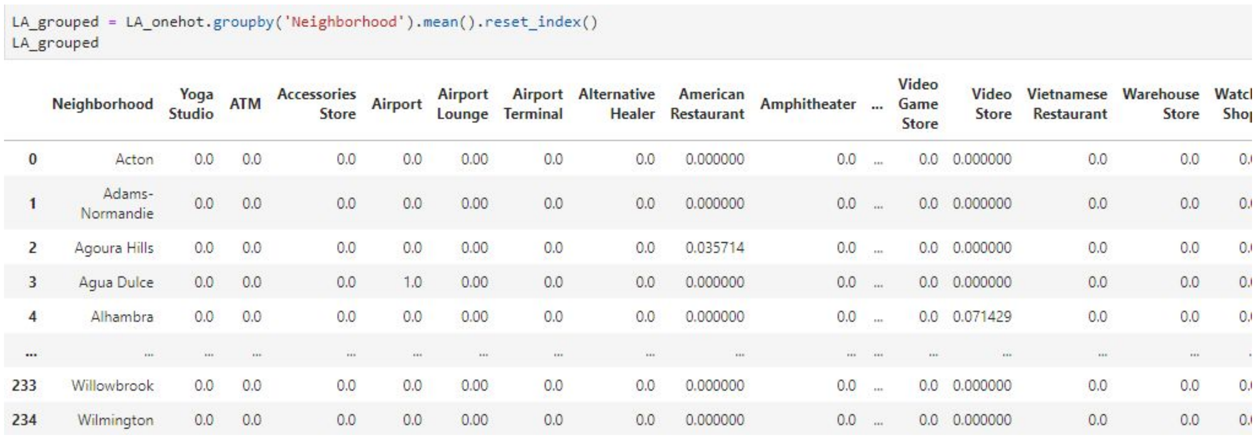
I decided to use K-means clustering machine learning algorithm to cluster the neighborhoods into hopefully meaningful clusters based on the venues.

In order to do that first I performed one hot encoding



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

# add neighborhood column back to dataframe
LA_onehot['Neighborhood'] = LA_venues['Neighborhood']

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

LA_onehot.head()
```

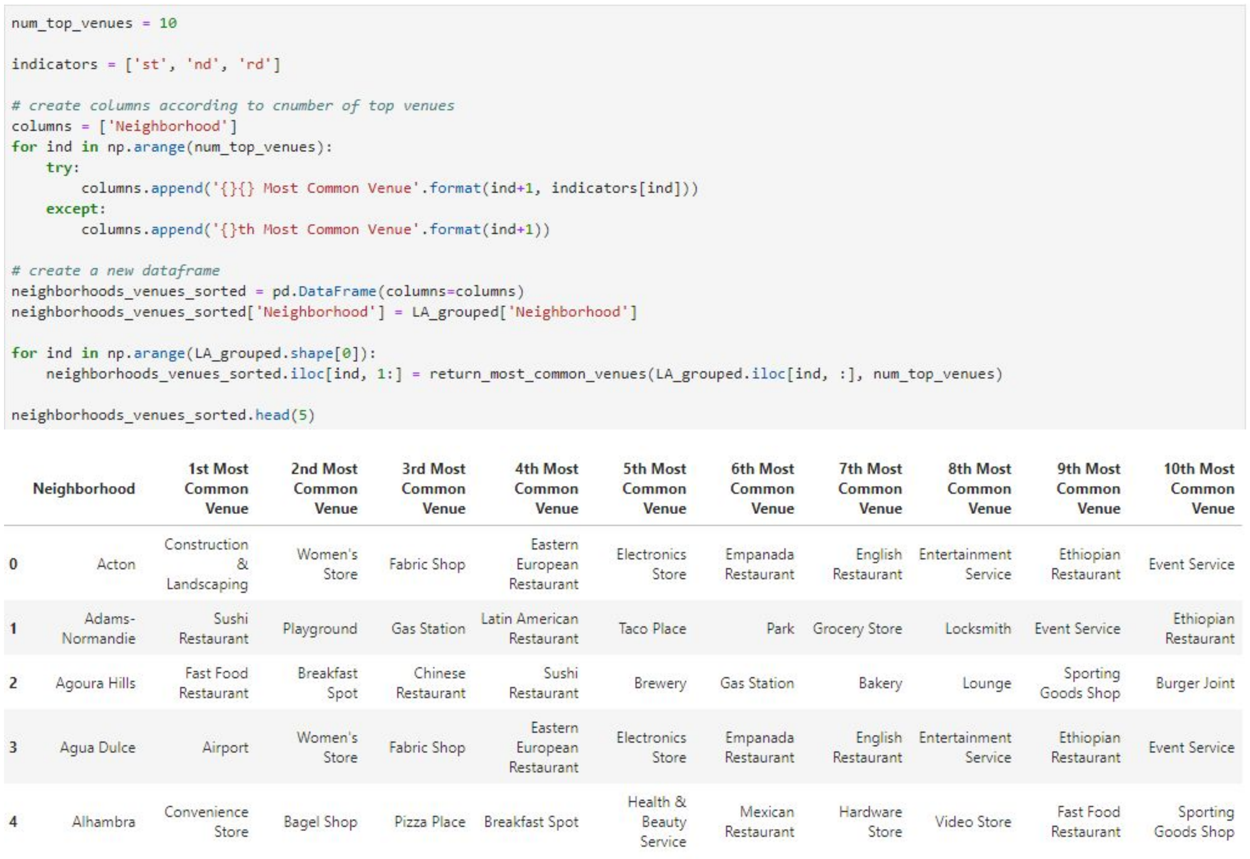
This is how the dataset looked like if the group it by the mean of the frequency of the occurrence of each category.



```
LA_grouped = LA_onehot.groupby('Neighborhood').mean().reset_index()
LA_grouped
```

	Neighborhood	Yoga Studio	ATM	Accessories Store	Airport	Airport Lounge	Airport Terminal	Alternative Healer	American Restaurant	Amphitheater	...	Video Game Store	Video Store	Vietnamese Restaurant	Warehouse Store	Watc Sho
0	Acton	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.000000	0.0	0.0	0.
1	Adams-Normandie	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.000000	0.0	0.0	0.
2	Agoura Hills	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.035714	0.0	...	0.0	0.000000	0.0	0.0	0.
3	Agua Dulce	0.0	0.0	0.0	1.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.000000	0.0	0.0	0.
4	Alhambra	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.071429	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
233	Willowbrook	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.000000	0.0	0.0	0.
234	Wilmington	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.000000	0.0	...	0.0	0.000000	0.0	0.0	0.

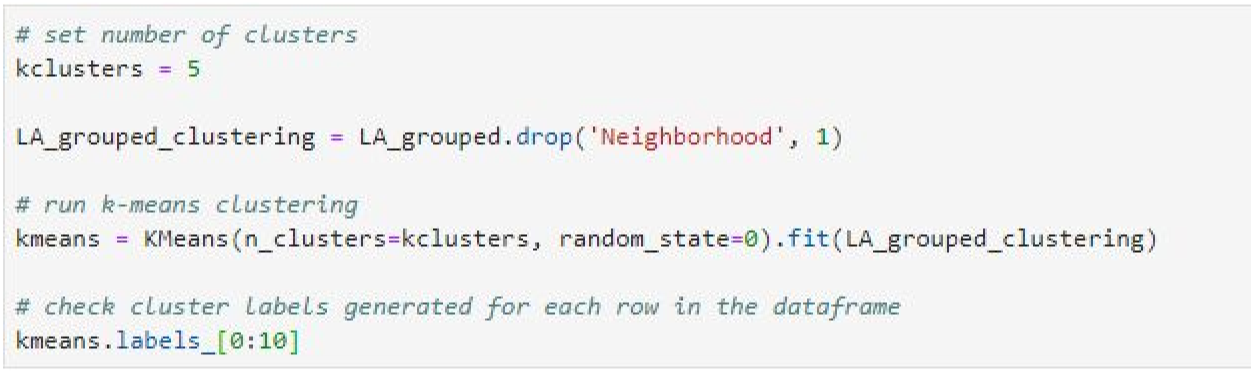
Fast forward I created a new dataframe that shows each neighborhood alongside with the top 10 most frequent venue categories.



```
num_top_venues = 10

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

# create columns according to cnumber 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'] = LA_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head(5)
```

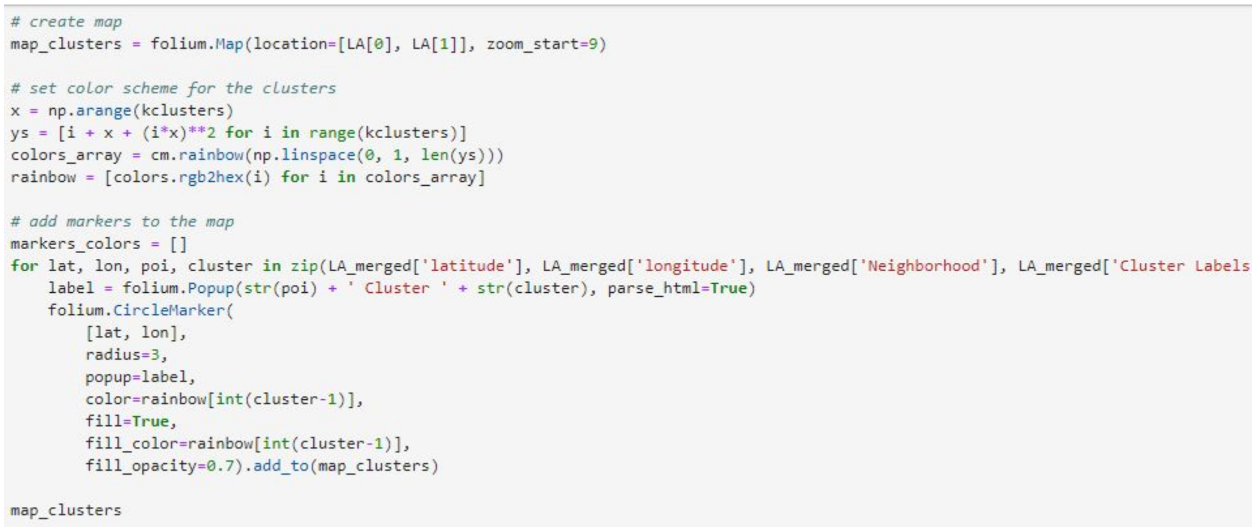
	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Acton	Construction & Landscaping	Women's Store	Fabric Shop	Eastern European Restaurant	Electronics Store	Empanada Restaurant	English Restaurant	Entertainment Service	Ethiopian Restaurant	Event Service
1	Adams-Normandie	Sushi Restaurant	Playground	Gas Station	Latin American Restaurant	Taco Place	Park	Grocery Store	Locksmith	Event Service	Ethiopian Restaurant
2	Agoura Hills	Fast Food Restaurant	Breakfast Spot	Chinese Restaurant	Sushi Restaurant	Brewery	Gas Station	Bakery	Lounge	Sporting Goods Shop	Burger Joint
3	Agua Dulce	Airport	Women's Store	Fabric Shop	Eastern European Restaurant	Electronics Store	Empanada Restaurant	English Restaurant	Entertainment Service	Ethiopian Restaurant	Event Service
4	Alhambra	Convenience Store	Bagel Shop	Pizza Place	Breakfast Spot	Health & Beauty Service	Mexican Restaurant	Hardware Store	Video Store	Fast Food Restaurant	Sporting Goods Shop

Clustering itself using K - means clustering

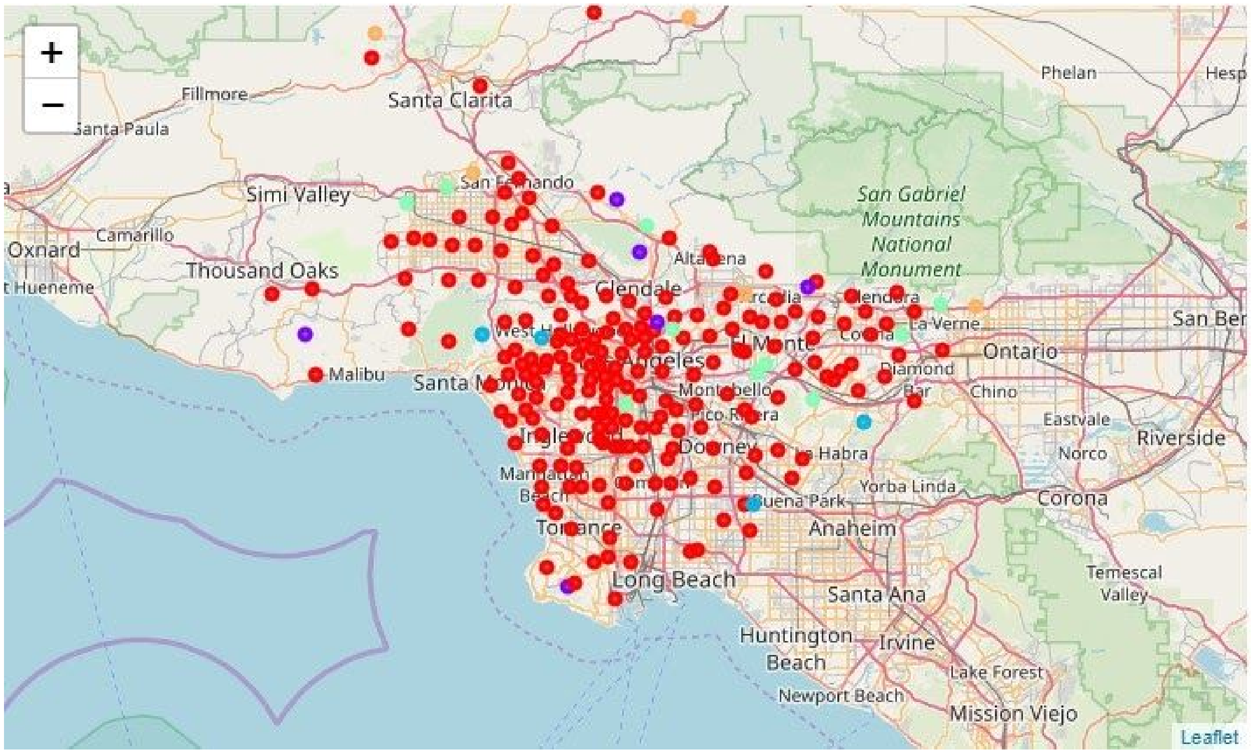


```
# set number of clusters
kclusters = 5

LA_grouped_clustering = LA_grouped.drop('Neighborhood', 1)

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

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

Visualising clusters using Folium.

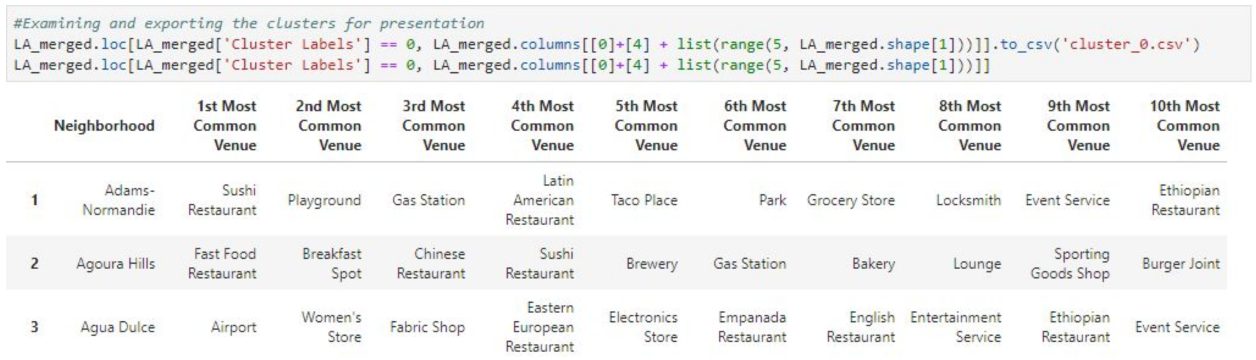


```
# create map
map_clusters = folium.Map(location=[LA[0], LA[1]], zoom_start=9)

# 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(LA_merged['latitude'], LA_merged['longitude'], LA_merged['Neighborhood'], LA_merged['Cluster Labels
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=3,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```



Clusters
0 - red
1 - purple
2 - blue
3 - green
4 - orange

Examining and exporting clusters into csv files for further use by the business team.



```
#Examining and exporting the clusters for presentation
LA_merged.loc[LA_merged['Cluster Labels'] == 0, LA_merged.columns[[0]+[4] + list(range(5, LA_merged.shape[1]))]].to_csv('cluster_0.csv')
LA_merged.loc[LA_merged['Cluster Labels'] == 0, LA_merged.columns[[0]+[4] + list(range(5, LA_merged.shape[1]))]]
```

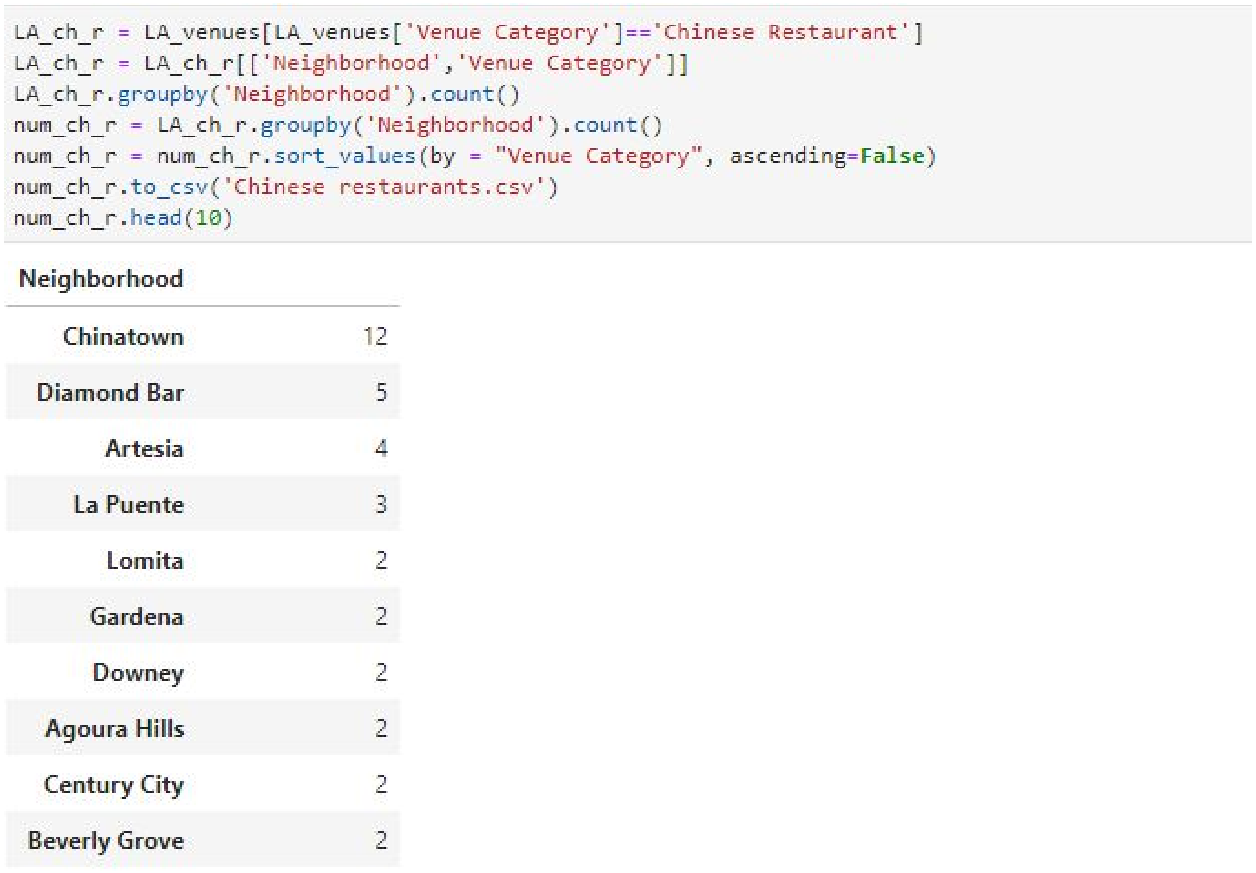
	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
1	Adams-Normandie	Sushi Restaurant	Playground	Gas Station	Latin American Restaurant	Taco Place	Park	Grocery Store	Locksmith	Event Service	Ethiopian Restaurant
2	Agoura Hills	Fast Food Restaurant	Breakfast Spot	Chinese Restaurant	Sushi Restaurant	Brewery	Gas Station	Bakery	Lounge	Sporting Goods Shop	Burger Joint
3	Agua Dulce	Airport	Women's Store	Fabric Shop	Eastern European Restaurant	Electronics Store	Empanada Restaurant	English Restaurant	Entertainment Service	Ethiopian Restaurant	Event Service

This type of information might be very useful for deciding a new location for a restaurant since neighborhoods can be clustered and understood as a whole.

Neighborhood by the number of Chinese restaurants.

Knowing how many Chinese restaurants are already built and running in a neighborhood in coup with other data can be of great use for the business team in determining the best location for the new restaurant.

I selected and grouped neighborhood in LA by the number of Chinese restaurants in them.

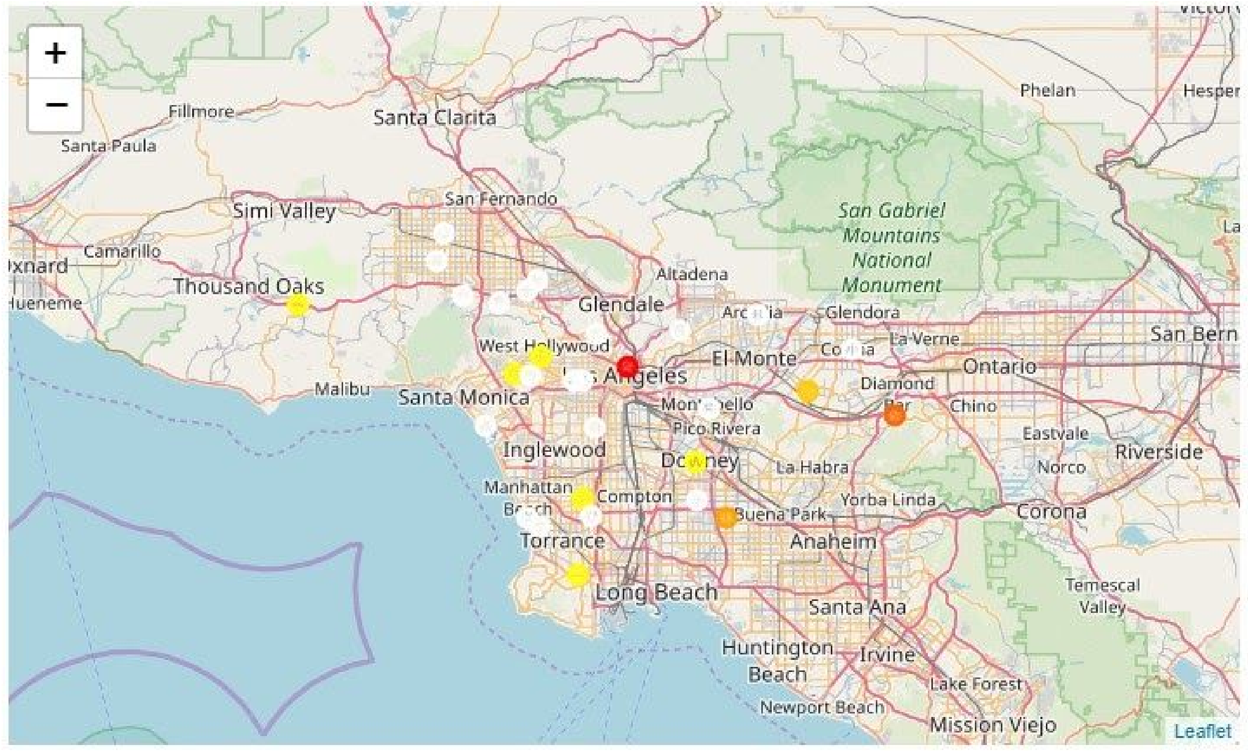


```python
LA_ch_r = LA_venues[LA_venues['Venue Category']=='Chinese Restaurant']
LA_ch_r = LA_ch_r[['Neighborhood','Venue Category']]
LA_ch_r.groupby('Neighborhood').count()
num_ch_r = LA_ch_r.groupby('Neighborhood').count()
num_ch_r = num_ch_r.sort_values(by = "Venue Category", ascending=False)
num_ch_r.to_csv('Chinese restaurants.csv')
num_ch_r.head(10)
```

Neighborhood	
Chinatown	12
Diamond Bar	5
Artesia	4
La Puente	3
Lomita	2
Gardena	2
Downey	2
Agoura Hills	2
Century City	2
Beverly Grove	2

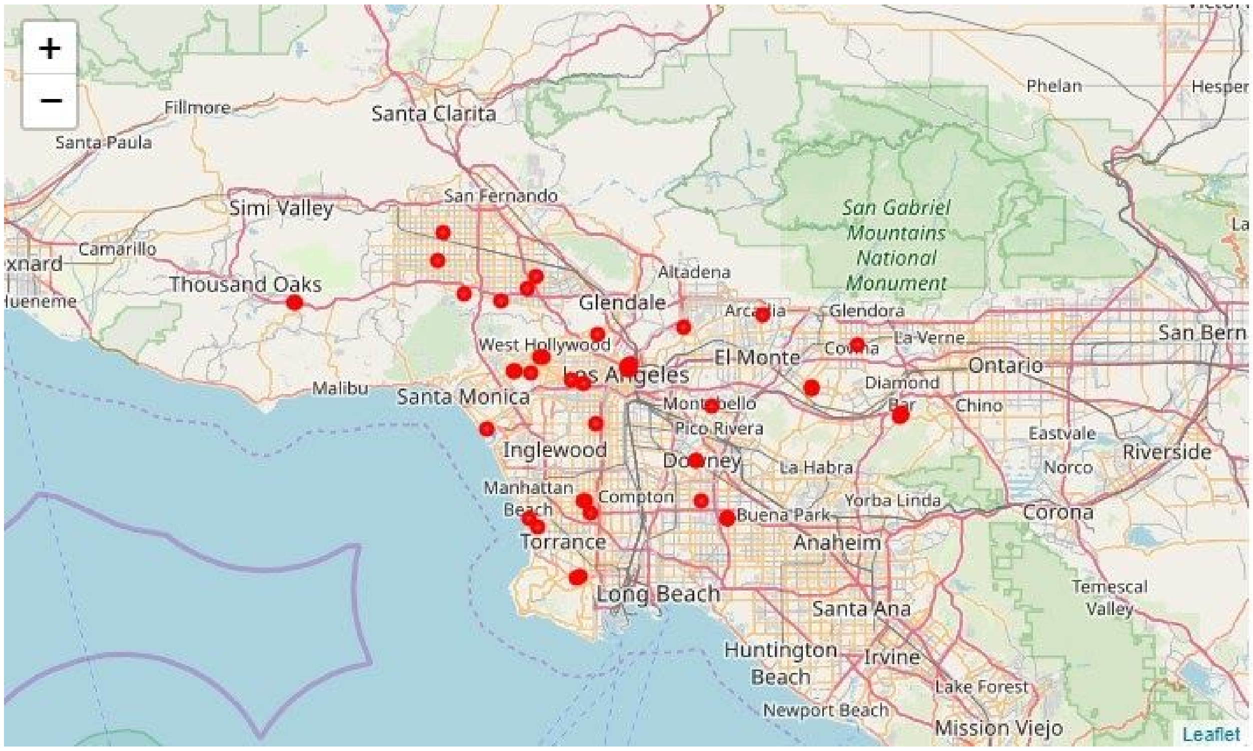
This table is also saved as a csv file for later use by the business team.

Then I proceeded to visualising the data on the map using Folium.



The restaurants themselves

Finally I added a visualization of all of the Chinese restaurants currently active in LA to get a better sense of the competition. This data is also saved in a CSV file for later use.

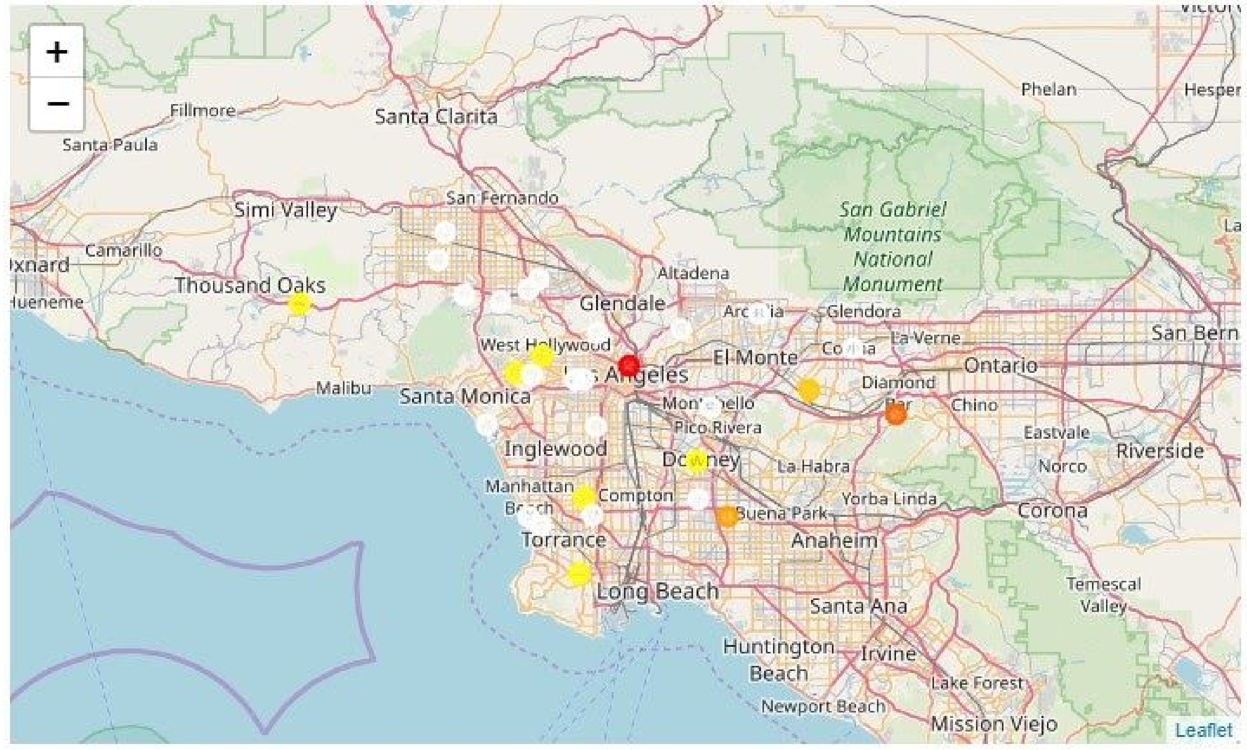


Results.

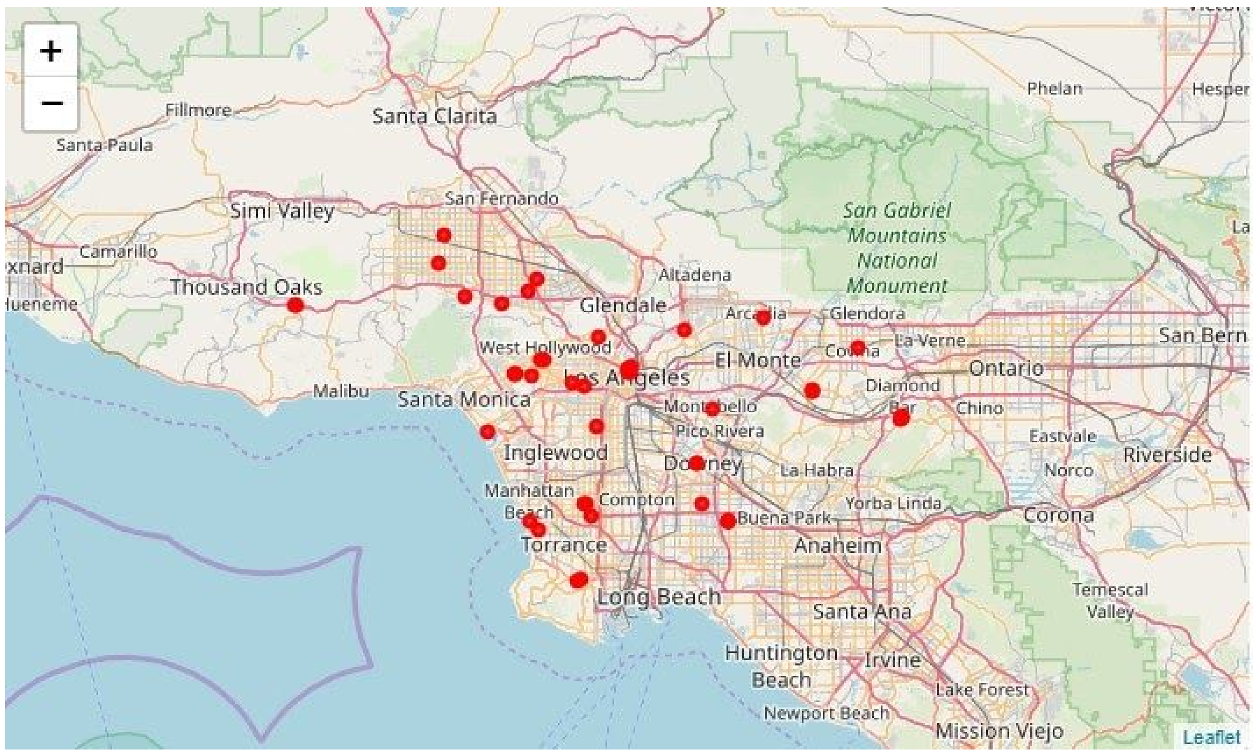
The goal of this analysis was to provide additional information for the CEO’s presentation on the issue.

The results are the following: 3 map visualisations regarding Chinese restaurants in LA and neighborhoods in LA alongside with CSV tables regarding the same information for later use.

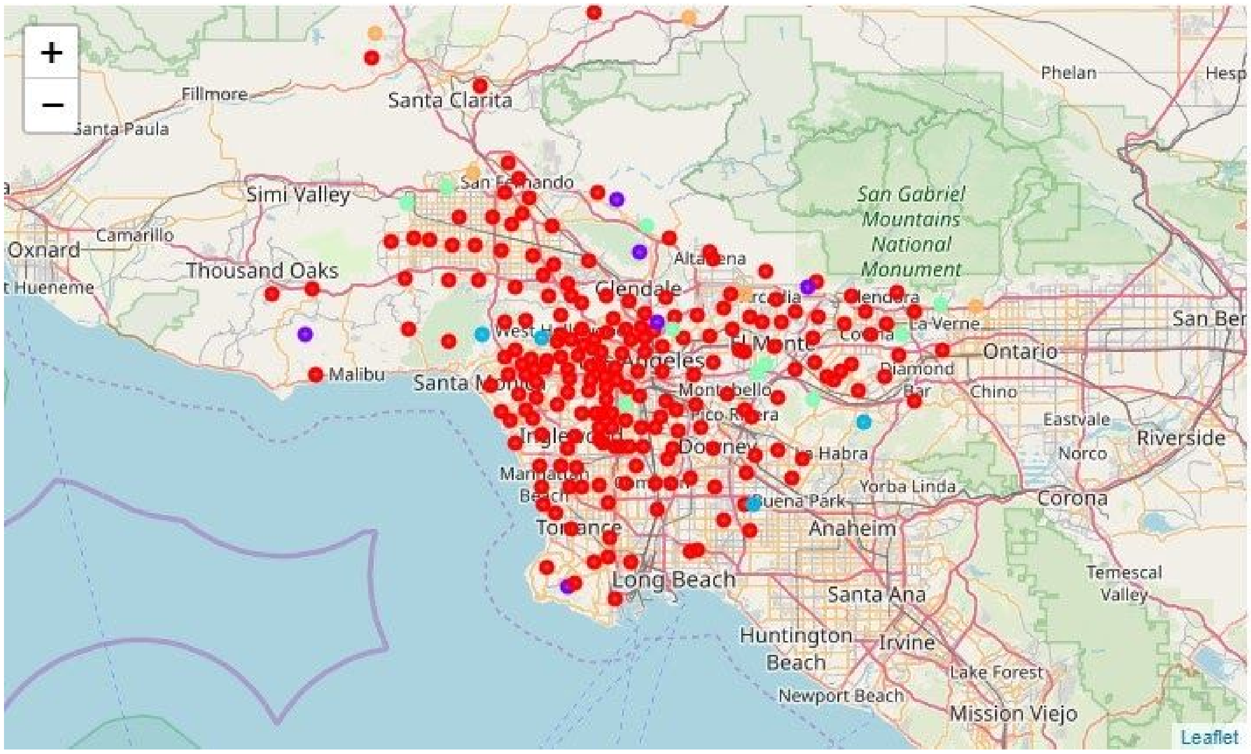
Neighborhoods in LA by the number of Chinese restaurants in them.



All Chinese restaurants in LA.



Neighborhoods in LA clustered into 5 clusters alongside with tables for each cluster for later use.



These tables and visualisations are meant to be used as additional information in deciding the best location for a new Chinese restaurant.

Discussion

In the background section I discussed how many factors are involved in deciding the best location for a venue. The results in this analysis are only complementary to other business criteria and add just a piece of the puzzle.

Based on my research I would highly recommend the business team to expand the time and budget allotted to the analysis which will allow using the state of the art data and methods such as Travel Time Factorization Model.

Or at least more time to incorporate additional analysis of income levels, real estate prices, history of success and failure of other Chinese restaurants in the city in order to better predict the success of a restaurant based on the location.

Conclusion

In a real life setting each project or a task has its limitations. Be it budget, time, or equipment. In this data analysis we saw what can be done within certain limitations to nethertheless add value to solving a business problem.