LADATA

May 29, 2020

0.0.1 Introduction

In this project we will use the geolospatial information of Los Angeles. The idea is to use the data to make suggestions regarding which districs are more suitable for opening a new restaurant. We will identify which districts have less restaurants than the rest, cluster them and make an appropriate suggestion. The fourquare database will be used to retrieve information for all the neighborhoods in our dataset. This step will be crucial when deciding to commence such an expensive procedure, like starting a new business. The same approach could be used to identify regions that are more suitable for opening new cafes. Of cource this is a simlified scenario. To address the question at each core, we should also take into account other factors, such as the average income in each neighborhood, the criminality levels, the average age of the citizens etc.. Obtaining this information for this project would be very hard to achieve, so we will restrict our analysis on the data we can retrieve from foursquare.

0.0.2 The dataset

The data for this study have been retrieved from: https://usc.data.socrata.com/dataset/Los-Angeles-Neighborhood-Map/r8qd-yxsr.

Let's first explore our data and plot a map of Los Angeles, highlighting with blue dots the neighborhoods of our dataset.

[160]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

#!conda install -c conda-forge geopy --yes # uncomment this line if you haven't␣
↪completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and␣
↪longitude values

import requests # library to handle requests
```

```
from pandas.io.json import json_normalize # tranform JSON file into a pandas␣
↪dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you␣
↪haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Libraries imported.
```

[161]:
```
with open('/home/christos/Downloads/LosAngelesNeighborhoodMap.geojson') as␣
↪json_data:
    la_data = json.load(json_data)
```

[162]:
```
neighborhoods_data = la_data['features']
```

[ ]:
```

```

[163]:
```
column_names = ['City', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)

for data in neighborhoods_data:
    neighborhood_name = data['properties']['name']
    neighborhood_lon = data['properties']['latitude']
    neighborhood_lat = data['properties']['longitude']

    neighborhoods = neighborhoods.append({'City': 'L.A.',
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon},␣
↪ignore_index=True)
```

[164]:
```
neighborhoods.sort_values('Neighborhood').head()
```

[164]:

	City	Neighborhood	Latitude	Longitude
0	L.A.	Acton	34.497355239240846	-118.16981019229348
1	L.A.	Adams-Normandie	34.031461499124156	-118.30020800000011

```
2  L.A.     Agoura Hills  34.146736499122795  -118.75988450000015
3  L.A.       Agua Dulce  34.504926999796837   -118.3171036690717
4  L.A.         Alhambra  34.085538999123571  -118.13651200000021
```

[165]:
```
address = 'Los Angeles'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geographical coordinate of L.A are {}, {}.'.format(latitude,␣
 ↪longitude))
```

```
The geographical coordinate of L.A are 34.0536909, -118.2427666.
```

[166]:
```
map_la = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, label in zip(neighborhoods['Latitude'],␣
 ↪neighborhoods['Longitude'], neighborhoods['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.9,
        parse_html=False).add_to(map_la)

map_la
```

[166]:
```
<folium.folium.Map at 0x7f509448e150>
```

0.0.3 Methodology

We are going to explore our data in order to reveal neighborhoods that are good targets for opening a new restaurant. We will identify which neighborhood have the least restaurants and make suggestions based on this observation. First, we can using basic plotting strategies, such as barplots, to compare the total number of restaurants in each neighborhoods, taking into account neighborhhods that have at least one restaurant. Next we will repeat the analysis taking into account all the available neighborhhods.

Since we are interested in the total number number of restaurants in each neighborhood and not on the type of the restaurant, we will have to modify the data reveived from foursquare. Foursquare originally returns the number of restaurants in each neighborhood, base on the type of the restaurant, i.e. Italian, Greek, Mexican etc. We will have to group our dataset based on the neighborhood name and estimate how many restaurants there are in total. This will also give us neighborhoods that do not have any restaurants (adding new information to the previous barplot).

Finally we are going to use the k-means clustering approach to make clusters of neighborhoods based on the venue information we have. Notice that we added a new venue that sums all the restaurants. You will notice that the map is not cleasr to make a proper decision. For this reason we are using a barplot to identify which labels correspond to neighborhoods with few or no restaurants. We can then suggest the appropriate neighborhoods based on this observation.

0.0.4 Analysis and results

[167]:
```
#time to use fousquare
CLIENT_ID = 'W3ZU1DV2Y5S1IAA1YLB4IZNNXCQXC4J4GTF1ON4DIVXUNJIV' # your␣
↪Foursquare ID
CLIENT_SECRET = 'FZB1RRL11JRXWRDHDBHZHSNSXQU4QPDKLFZ5LB4QFZUJII2O' # your␣
↪Foursquare Secret
VERSION = '20180605' # Foursquare API versionb
```

[168]:
```
#explore neighborhoods in Toronto


def getNearbyVenues(names, latitudes, longitudes,LIMIT=100, radius=500):

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

        # 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']
```

```python
        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item␣
↪in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

[169]:
```python
la_venues = getNearbyVenues(names=neighborhoods['Neighborhood'],
                                   latitudes=neighborhoods['Latitude'],
                                   longitudes=neighborhoods['Longitude']
                                  )
```

[ ]:
```python

```

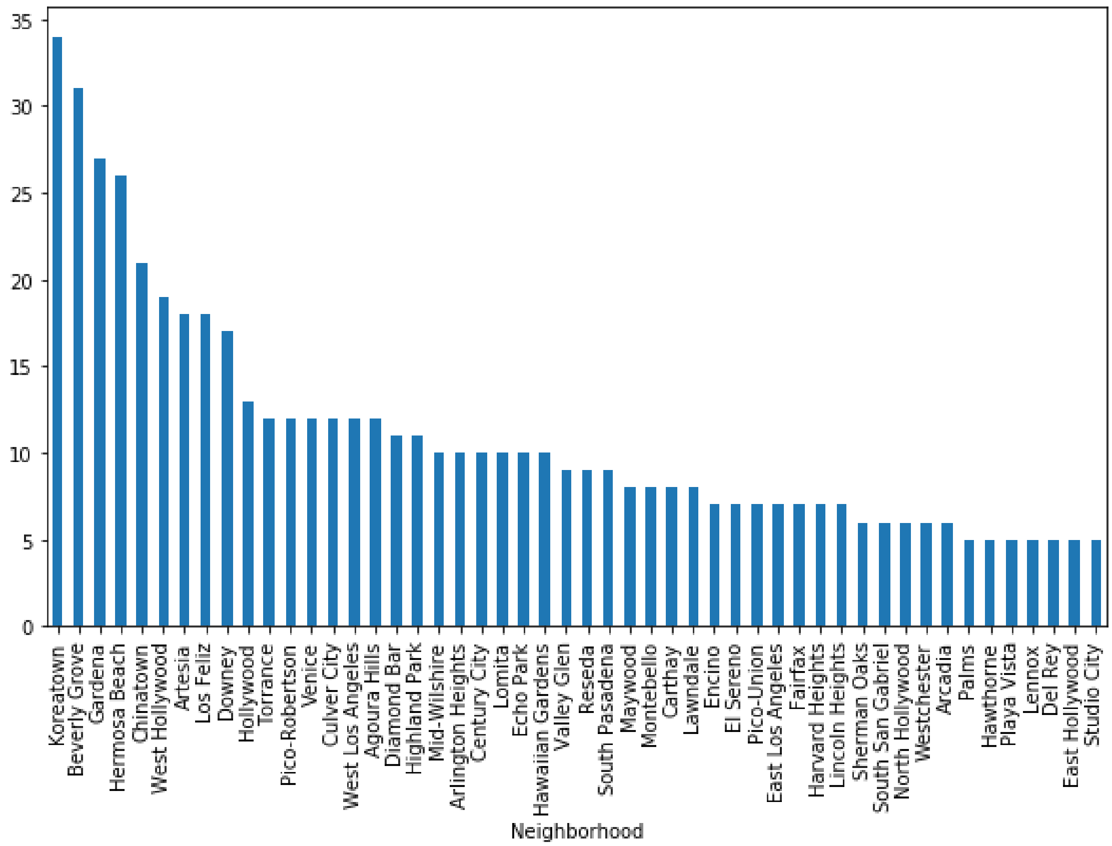
[170]:
```python
##keep only restaurants
la_venues_forfood=la_venues[la_venues['Venue Category'].str.
↪contains("Restaurant")].reset_index(drop=True)
```

[ ]:
```python

```

[153]:
```python
#take total number of restaurants in each neighborhood
#doing this we ignore neighborhoods with no restaurants
```

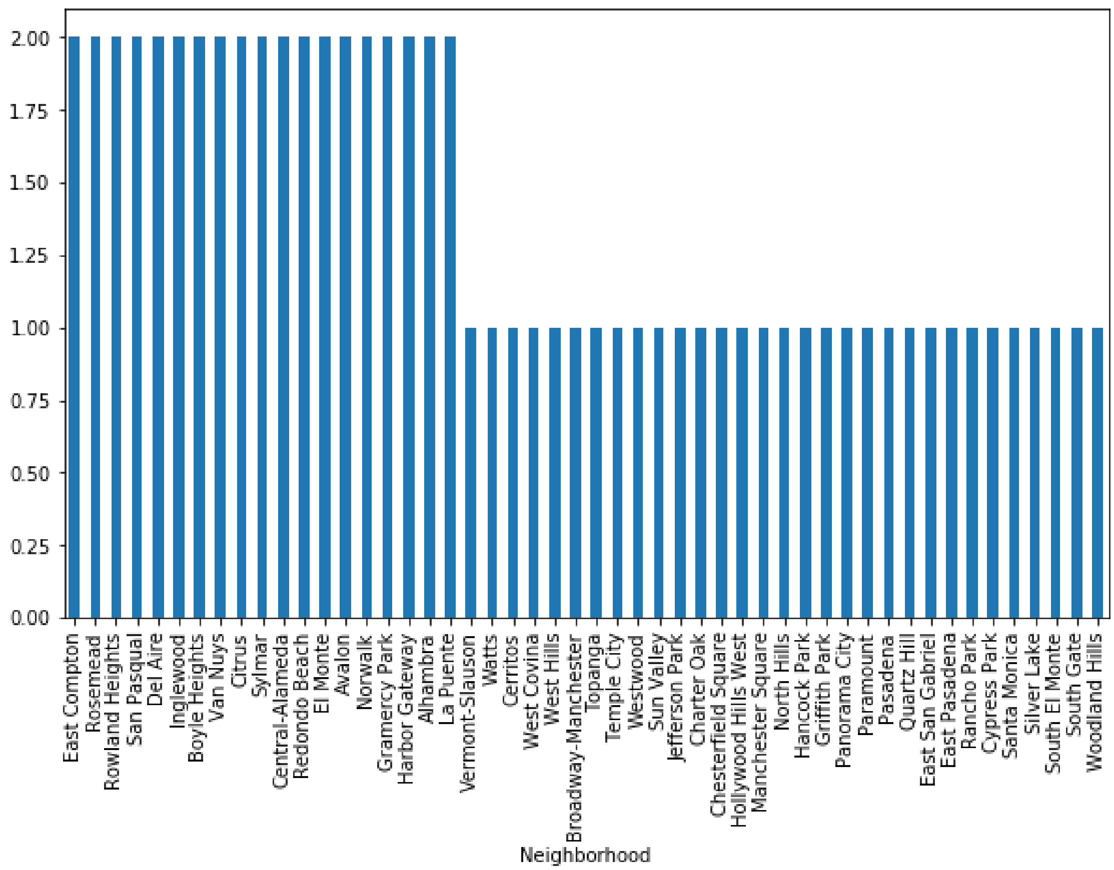
[171]:
```python
#plot the top 50 neighborhoods with the most restaurants

la_venues_forfood.groupby('Neighborhood').count()['Venue Category'].
↪sort_values(ascending=False).head(50).plot(kind='bar', figsize=(10, 6))
```

[171]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5089779f90>



[172]:
```
#plot the bottom 50 neighborhoods with the least restaurants

la_venues_forfood.groupby('Neighborhood').count()['Venue Category'].
↪sort_values(ascending=False).tail(50).plot(kind='bar', figsize=(10, 6))
```

[172]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5089804c10>



0.0.5 From the plots above neighborhoods with no restaurants at all are missing. Lets try to include them in our analysis.

[ ]:

```

```

[173]:

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

# add neighborhood column back to dataframe
la_hot['NeighborhoodName'] = la_venues['Neighborhood']

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

[174]:

```
#la_hot.iloc[353:429,:][la_hot.columns[la_hot.columns.str.
↪contains("Restaurant")]].sum(axis=1).sum()
```

[175]:
```
la_hotforfood=la_hot[la_hot.columns[la_hot.columns.str.contains("Restaurant")]]
la_hotforfood['NeighborhoodName']=la_hot['NeighborhoodName']
fixed_columns2 = [la_hotforfood.columns[-1]] + list(la_hotforfood.columns[:-1])
la_hotforfood = la_hotforfood[fixed_columns2]
```

```
/home/christos/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```

```

[176]:
```
la_hotforfood['TotalRestaurants']=la_hotforfood[la_hotforfood.
↪columns[la_hotforfood.columns.str.contains("Restaurant")]].sum(axis=1)
```

[ ]:
```

```

[177]:
```
la_hotforfoodTotal=la_hotforfood.groupby('NeighborhoodName').
↪sum()[['TotalRestaurants']]
```

[178]:
```
la_hotforfoodTotal.index.name = None
la_hotforfoodTotal['NeighborhoodName']=la_hotforfoodTotal.index.tolist()
```

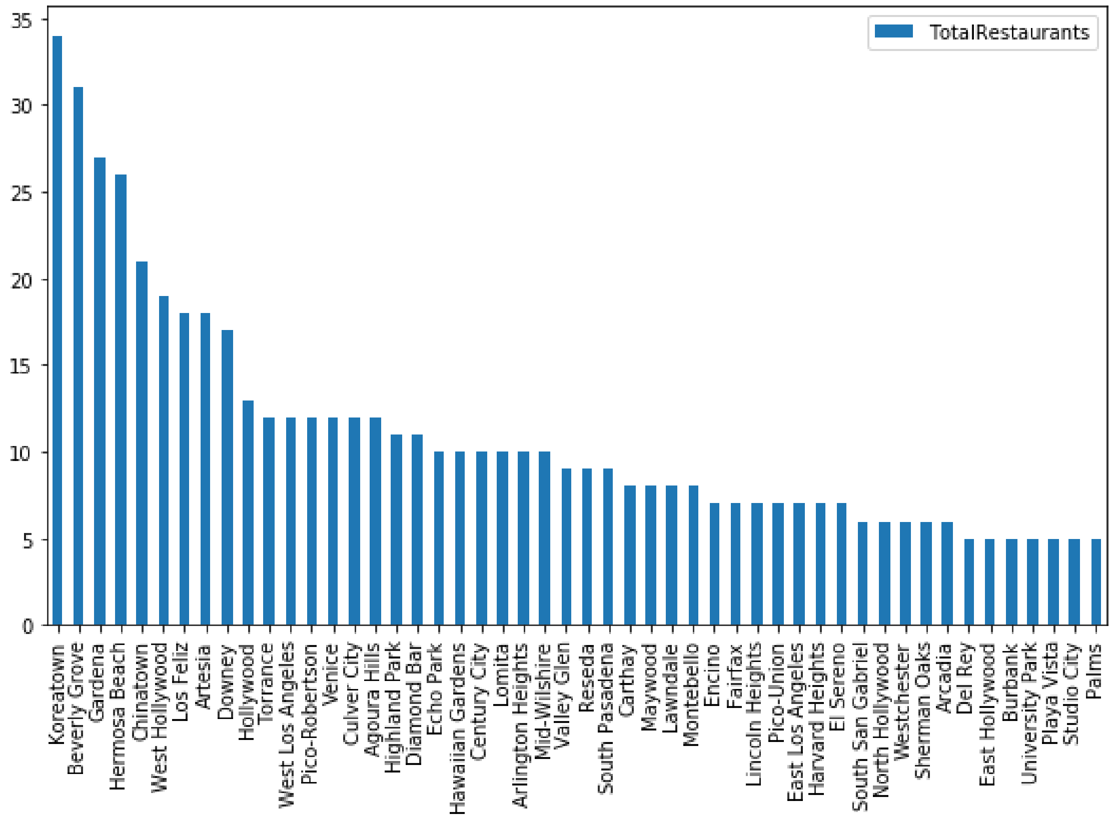
[179]:
```
la_hotforfoodTotal.head(10)
```

[179]:

	TotalRestaurants	NeighborhoodName
Acton	0	Acton
Adams-Normandie	3	Adams-Normandie
Agoura Hills	12	Agoura Hills
Agua Dulce	0	Agua Dulce
Alhambra	2	Alhambra
Alondra Park	0	Alondra Park
Altadena	0	Altadena
Arcadia	6	Arcadia
Arleta	0	Arleta
Arlington Heights	10	Arlington Heights

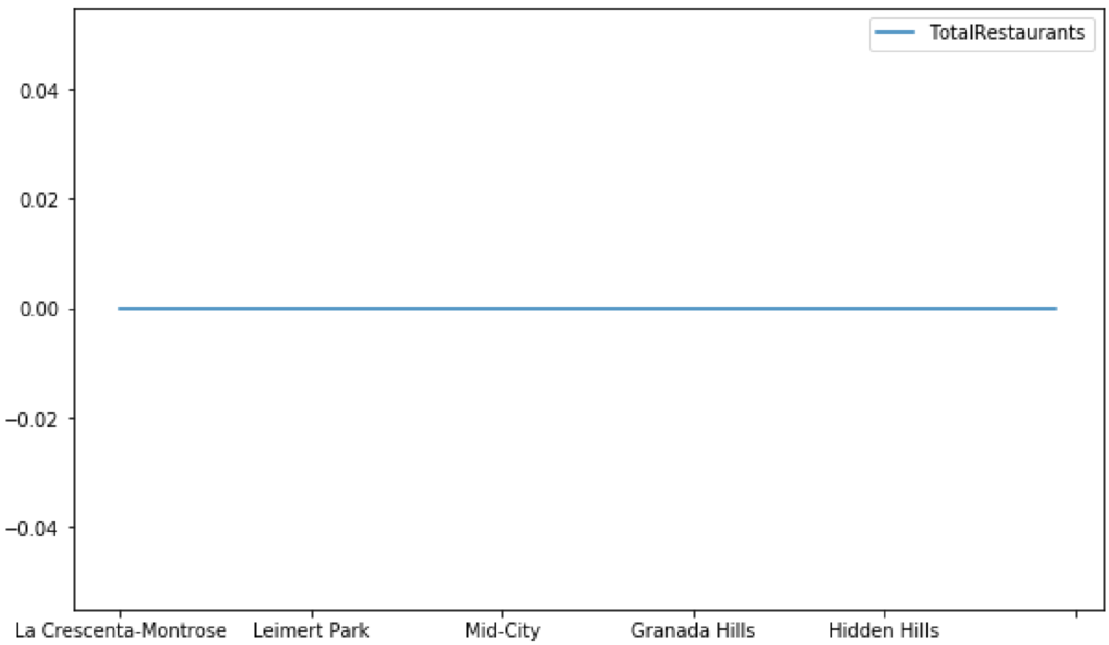
[180]:
```
#repeat the barplot analysis including neighborhoods that do not have any␣
↪restaurants

la_hotforfoodTotal.sort_values(by='TotalRestaurants',ascending=False).head(50).
↪plot(kind='bar',figsize=(10, 6))
```

[180]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5089183650>



[181]:
```
la_hotforfoodTotal.sort_values(by='TotalRestaurants',ascending=False).tail(50).
↪plot(kind='line',figsize=(10, 6))
```

[181]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50940d9f10>



0.0.6 We see that there are many ares with no restaurants at all. Lets try to cluster the neighborhoods.

[182]:
```
#keep no-restaurant venues in a separate dataframe
la_hotnofood=la_hot[la_hot.columns[la_hot.columns.str.
 ↪contains("Restaurant")==False]]
```

[183]:
```
# now we have two dataframes, one that has information only for the␣
 ↪restaurants,
#and one that have information for the rest venues

print(la_hotnofood.shape,la_hotforfood.shape)
```

```
(2958, 264) (2958, 58)
```

[184]:
```
#let's merge the two datasets
# now we removed the differents types of restaurants and kept only the total␣
 ↪number of restaurants in each
#neighborhhod
la_hotTotal=pd.
 ↪concat([la_hotforfood[['TotalRestaurants']],la_hotnofood],axis=1, sort=False)
```

[ ]:
```
```

[185]:
```
la_grouped = la_hotTotal.groupby('NeighborhoodName').mean()
```

[186]:
```
#perform clustering

# set number of clusters
kclusters = 10

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_.shape
```

[186]:
```
(239,)
```

[195]:
```
neighborhoods.index=neighborhoods['Neighborhood']
indexx=la_grouped.index.tolist()
neighborhoods_new=neighborhoods.loc[indexx]

neighborhoods_new.insert(0, 'Cluster Labels', kmeans.labels_)
```

[196]:
```
neighborhoods_new.head()
```

[196]:
```
                  Cluster Labels  City     Neighborhood           Latitude  \
Neighborhood
Acton                          5  L.A.            Acton  34.497355239240846
Adams-Normandie                0  L.A.  Adams-Normandie  34.031461499124156
Agoura Hills                   0  L.A.     Agoura Hills  34.146736499122795
Agua Dulce                     8  L.A.       Agua Dulce  34.504926999796837
Alhambra                       8  L.A.         Alhambra  34.085538999123571

                           Longitude
Neighborhood
Acton             -118.16981019229348
Adams-Normandie   -118.30020800000011
Agoura Hills      -118.75988450000015
Agua Dulce         -118.3171036690717
Alhambra          -118.13651200000021
```

[197]:
```
import matplotlib.cm as cm
import matplotlib.colors as colors

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 = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(neighborhoods_new['Latitude'],␣
↪neighborhoods_new['Longitude'], neighborhoods_new['Neighborhood'],␣
↪neighborhoods_new['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)

map_clusters
```

[197]:

```
<folium.folium.Map at 0x7f5088fcc3d0>
```

0.0.7 We can clearly say that we cannot make a suggestion based on this map. Let's try again keeping only information regarding the restaurant number in each neighborhood.

[198]:

```
la_grouped_resta=la_grouped[['TotalRestaurants']]
```

[199]:

```
la_grouped_resta.head()
```

[199]:

NeighborhoodName	TotalRestaurants
Acton	0.000000
Adams-Normandie	0.333333
Agoura Hills	0.428571
Agua Dulce	0.000000
Alhambra	0.153846

[200]:

```
#perform clustering

# set number of clusters
kclusters = 10

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

```
# check cluster labels generated for each row in the dataframe
kmeans.labels_.shape
```

[200]: (239,)

[201]:
```
indexx=la_grouped.index.tolist()
neighborhoods_resta=neighborhoods.loc[indexx]

neighborhoods_resta.insert(0, 'Cluster Labels', kmeans.labels_)
```

[202]:
```
import matplotlib.cm as cm
import matplotlib.colors as colors

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 = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(neighborhoods_resta['Latitude'],␣
 ↪neighborhoods_resta['Longitude'], neighborhoods_resta['Neighborhood'],␣
 ↪neighborhoods_resta['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)

map_clusters
```

[202]: <folium.folium.Map at 0x7f5088db76d0>

0.0.8 Slightly better. We have better cluster now, but still not clear. Let's try to make a barplot to see wch clusters correspond to neighborhoods with few restaurants.

[203]:
```
la_grouped_resta['Neighborhood']=la_grouped_resta.index.values.tolist()
la_grouped_resta2=la_grouped_resta.reset_index()
la_grouped_resta2.drop('NeighborhoodName',axis=1, inplace=True)
la_grouped_resta2.drop('Neighborhood',axis=1, inplace=True)
```

```
/home/christos/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[204]:
```
neighborhoods_resta.index.name=None
```

[205]:
```
neighborhoods_resta2=neighborhoods_resta.reset_index()
```

[206]:
```
neighborhoods_resta2.drop('index', axis=1, inplace=True)
```

[207]:
```
neighborhoods_resta2.head()
```

[207]:
```
   Cluster Labels  City     Neighborhood            Latitude  \
0               0  L.A.            Acton  34.497355239240846
1               1  L.A.  Adams-Normandie  34.031461499124156
2               5  L.A.     Agoura Hills  34.146736499122795
3               0  L.A.       Agua Dulce  34.504926999796837
4               6  L.A.         Alhambra  34.085538999123571

            Longitude
0  -118.16981019229348
1  -118.30020800000011
2  -118.75988450000015
3   -118.3171036690717
4  -118.13651200000021
```

[208]:
```
la_grouped_resta2.head()
```

[208]:
```
   TotalRestaurants
0          0.000000
1          0.333333
2          0.428571
3          0.000000
```

```
4          0.153846
```

[209]:
```
final_dataset=pd.concat([la_grouped_resta2,neighborhoods_resta2],axis=1,␣
↪sort=False)
```

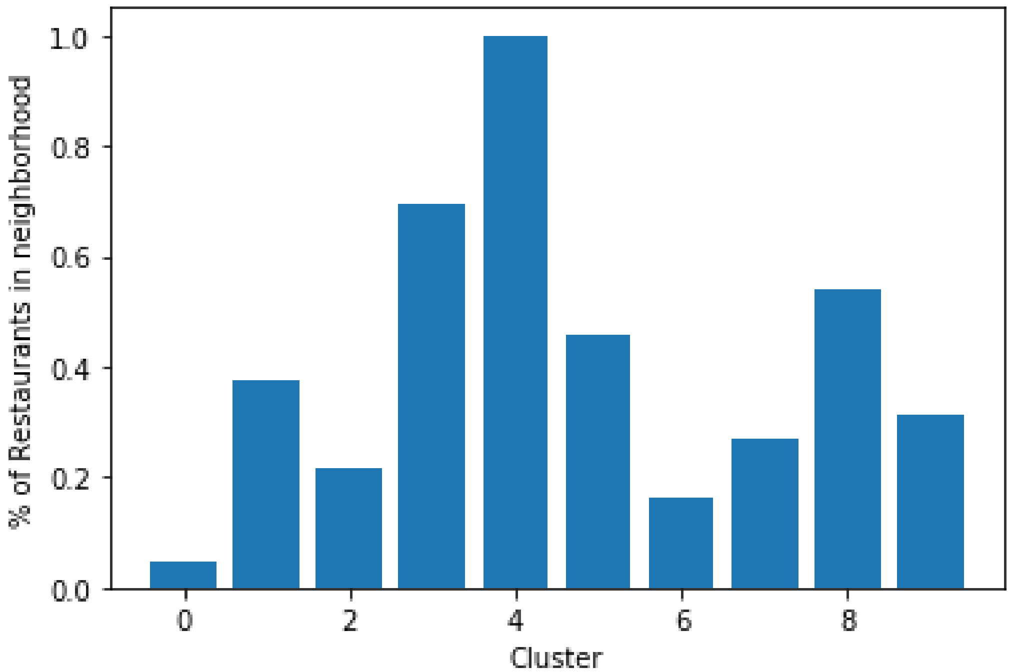
[211]:
```
final_dataset.head()
```

[211]:
```
   TotalRestaurants  Cluster Labels  City     Neighborhood  \
0          0.000000               0  L.A.            Acton
1          0.333333               1  L.A.  Adams-Normandie
2          0.428571               5  L.A.     Agoura Hills
3          0.000000               0  L.A.       Agua Dulce
4          0.153846               6  L.A.         Alhambra

             Latitude            Longitude
0  34.497355239240846  -118.16981019229348
1  34.031461499124156  -118.30020800000011
2  34.146736499122795  -118.75988450000015
3  34.504926999796837   -118.3171036690717
4  34.085538999123571  -118.13651200000021
```

[212]:
```
import matplotlib.pyplot as plt
```

[213]:
```
plt.bar(x=final_dataset['Cluster␣
↪Labels'],height=final_dataset['TotalRestaurants'])
plt.xlabel('Cluster')
plt.ylabel('% of Restaurants in neighborhood')
```

[213]:
```
Text(0, 0.5, '% of Restaurants in neighborhood')
```



0.0.9 We can say that neighborhoods with cluster labels 0 and 6 do not have may restaurants. Let's view these neighborhoods.

[214]:
```
final_dataset.loc[(final_dataset['Cluster Labels']==0) |␣
↪(final_dataset['Cluster Labels']==6) ,:].head()
```

[214]:
```
   TotalRestaurants  Cluster Labels  City  Neighborhood           Latitude  \
0          0.000000               0  L.A.         Acton  34.497355239240846
3          0.000000               0  L.A.    Agua Dulce  34.504926999796837
4          0.153846               6  L.A.      Alhambra  34.085538999123571
5          0.000000               0  L.A.  Alondra Park  33.889617004889644
6          0.000000               0  L.A.      Altadena  34.193870502232173

            Longitude
0  -118.16981019229348
3   -118.3171036690717
4  -118.13651200000021
5  -118.33515598608159
6  -118.13623898201556
```

0.0.10 Discussion

Using the k-means clustering approach we identied neighborhoods that cluster together, based only on the number of restaurants in each neighborhood. We found that neighborhoods labeled with 0 and 6 do not have many restaurants and subsequently would be the best options to start a new restaurant.

0.0.11 Conclusion

Trying to make suggestion about carrer opportunities is not easy. Here, we combined basic plotting startegies with clustering methods to obtain a basic intuition about which neighborhood in L.A is appropriate for opening a new restaurant. We based our analysis only on the number of restaurants in each area. To complete this analysis many more parameters must be taken into account. This would be just the first step trying to answer a real-world reseach question.