Introduction: Business Problem

Curitiba is one of the most important cities in the southern part of Brazil. As capital of the state of Paraná, the city has more than 1,800,000 habitants and covers an area of 430.9 km². It is the 7th largest Brazilian city and 4th largest in the Southern Cone (the south part of South America). The city has the largest population and the largest economy in Southern Brazil.[5] The urban area of Curitiba is looked after by 26 local governments and has 3,335,588 people living there.

Curitiba is a city in south Brazil, capital of Paraná. It has 75 neighborhoods, divided into 9 administrative regions. The city itself has a population of 1.9 million people. It's continuously built-up urban area, is the eighth largest city in Brazil.

Curitiba is considered a leading regional city, with a strong market, culture, art, finance and education areas. Its industrial district hosts larges national and international companies, over last decades the district is continuously growing because of the city's important role in government and commercial business.

Therefore, in this project I would like to help tourists or even residents with an analysis of the rated restaurants distribution and finally with a quick guide of the best places and region to eat for cheap in Curitiba. Some questions we are looking to answer:

• Which areas/neighborhoods have more restaurants in Curitiba?
• What are the best-rated restaurants to eat for cheap in Curitiba?
• Is it possible to define the best location to eat for cheap in Curitiba?

Target audience:

People interested on eating well for less, tourists or residents who looks for a guide of great food for cheap. Business Analyst, who wish to analyze the neighborhoods of Curitiba using python, Jupiter notebook and some machine learning techniques.

Data

The following data will be used for this purpose:

- 1st Data: https://pt.wikipedia.org/wiki/Lista_de_bairros_de_Curitiba (https://pt.wikipedia.org/wiki/Lista_de_bairros_de_Curitiba)
  The list of Curitiba neighborhoods, with area, population and average net income;
- 2nd Data: https://nominatim.openstreetmap.org/ (https://nominatim.openstreetmap.org/)
  Coordinates from each neighborhood;
- 3rd Data: https://developer.foursquare.com/ (https://developer.foursquare.com/)
  Restaurants lists database, containing Category, ID and coordinates;
- 4rd Data: https://developer.foursquare.com/ (https://developer.foursquare.com/)
  Restaurants ratings, likes, tips and price;

But before dealing with the data, lets first install the necessary librariesfor this project, in case they weren't installed before:

In [ ]:
```
!pip install --user wikipedia
!pip install --user folium
!pip install --user geopy
!pip install --user geopandas
!pip install --user geojson
!pip install --user yellowbrick
!pip install --user pyproj
```

And now import them:

In [2]:
```
import pandas as pd
from pandas.io.json import json_normalize # tranform JSON file into a pandas dat

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

#import wikipedia as wp
import requests
from bs4 import BeautifulSoup
import numpy as np # library to handle data in a vectorized manner
import json # library to handle JSON files
from geopy.geocoders import Nominatim # convert an address into latitude and lon
#import geopandas as gpd

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

# map rendering library
import folium
from folium import plugins
from folium.plugins import HeatMap

# KMeans
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

# Extras
import pickle
from functools import reduce
from pyproj import Geod

print('Libraries imported.')
```

```
Libraries imported.
```

The first data is a Wikipedia page about Curitiba neighborhoods. Neighborhoods are spitted into nine boroughs, each one in one table, that contains information of population, households, average income and area. We will scrape the page and create a data frame consisting of seven columns: Neighborhood, Area, Men, Women, Total, Households, Avg. Income. With the Area and Total (total population) columns we can calculate Population Density and create another column;

Now let's scrape our first data from wikipedia and parse using BeautifulSoup:

1st Data - Wikipedia

In [4]:
```
# fetching Wikipedia data
data = requests.get('https://pt.wikipedia.org/wiki/Lista_de_bairros_de_Curitiba'
```

In [5]:
```
# parsing data using BeautifulSoup
soup = BeautifulSoup(data, 'html.parser')
```

In [6]:
```
# find tables
tables = soup.find_all('table') # in html table is represented by the tag <table
```

In [ ]:
```
# create dataframe with Curitiba Neighborhoods

temp_content=[]

for index,table in enumerate(tables):
    if 1 < index < 11:
        temp = pd.read_html(str(table), flavor='bs4',thousands=' ')[0]
        temp = temp.drop([0,1,2])
        temp_content.append(temp)


temp_content = pd.concat(temp_content)
curitiba_df = pd.DataFrame(temp_content)
curitiba_df.reset_index(drop=True, inplace=True)
column_names = ['Neighborhood', 'Area', 'Men', 'Women', 'Total', 'Households','A
curitiba_df.columns = column_names
```

In [20]:
```
# convert numbers stored as str

for i in range(1,7):
    curitiba_df.iloc[:,i] = curitiba_df.iloc[:,i].str.replace(',', '')
    curitiba_df.iloc[:,i] = curitiba_df.iloc[:,i].str.split(' ')
    curitiba_df.iloc[:,i] = curitiba_df.iloc[:,i].agg(lambda x: ' '.join(map(str
    curitiba_df.iloc[:,i] = pd.to_numeric(curitiba_df.iloc[:,i], errors='coerce'
    if i == 6 or i == 1:
        curitiba_df.iloc[:,i] = curitiba_df.iloc[:,i]/100
```

In [ ]:
```
# calculate population density

curitiba_df['Population Density'] = curitiba_df['Total']/curitiba_df['Area']
```

Let's take a look into our dataframe

In [16]:
```
curitiba_df
```

Out[16]:

	Neighborhood	Area	Men	Women	Total	Households	Avg. Income	Population Density
0	Ganchinho	11.20	3667	3658	7325	1921	767.35	654.017857
1	Sitio Cercado	11.12	50631	51779	102410	27914	934.95	9209.532374
2	Umbará	22.47	7280	7315	14595	17064	908.70	649.532710
3	Abranches	4.32	5463	5702	11165	3154	1009.67	2584.490741
4	Atuba	4.27	6156	6476	12632	3627	1211.60	2958.313817
5	Bacacheri	6.98	10762	12344	23106	7107	3029.00	3310.315186
6	Bairro Alto	7.02	20244	21789	42033	12071	1211.60	5987.606838
7	Barreirinha	3.73	8079	8942	17021	5024	1272.18	4563.270777
8	Boa Vista	5.14	13677	15714	29391	9212	1817.40	5718.093385
9	Cachoeira	3.07	3811	3927	7738	2091	908.70	2520.521173
10	Pilarzinho	7.13	13358	14549	27907	7883	1211.60	3914.025245

In order to save our databases into files for later use, we are going to use pickle.

[https://www.synopsys.com/blogs/software-security/python-pickling/ (https://www.synopsys.com/blogs/software-security/python-pickling/)](https://www.synopsys.com/blogs/software-security/python-pickling/)

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it’s the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. The pickled byte stream can be used to re-create the original object hierarchy by unpickling the stream. This whole process is similar to object serialization in Java or .Net.

In [154]:
```
# save curitiba_df to pickle

with open('curitiba_df.pkl', 'wb') as f:
        pickle.dump(curitiba_df, f)
```

In [3]:
```
# load curitiba_df from pickle

with open('curitiba_df.pkl', 'rb') as f:
        curitiba_df = pickle.load(f)
```

2st Data - Nominatim

Then, we will be using the Geocoder Nominatim python package to retrieve the Neighborhoods coordinates. This is our second database and we stored it with pickle and joined into our main curitiba_df database.

In [69]:

```
# Use geopy library to get the latitude and longitude values of Curitiba Neighbo

location = [x for x in curitiba_df['Neighborhood'].unique().tolist()
            if type(x) == str]
latitude = []
longitude =  []
for i in range(0, len(location)):
    try:
        address = location[i] + ', Curitiba, PR, Brazil'
        geolocator = Nominatim(user_agent="curitiba_dataset")
        loc = geolocator.geocode(address)
        latitude.append(loc.latitude)
        longitude.append(loc.longitude)
        print('The geographical coordinate of {} are {}, {}.'.format(location[i]
    except:
        # in the case the geolocator does not work, then add nan element to list
        # to keep the right size
        latitude.append(np.nan)
        longitude.append(np.nan)

# create a dataframe with the location, latitude and longitude
df_cwb_coord = pd.DataFrame({'Neighborhood':location,
                    'Latitude': latitude,
                    'Longitude':longitude})
```

```
The geographical coordinate of Ganchinho are -25.5720763, -49.2636674.
The geographical coordinate of Sitio Cercado are -25.5427012, -49.2691056.
The geographical coordinate of Umbará are -25.5681693, -49.2856994.
The geographical coordinate of Abranches are -25.3614742, -49.272054.
The geographical coordinate of Atuba are -25.3875003, -49.2066058.
The geographical coordinate of Bacacheri are -25.3968497, -49.2344563.
The geographical coordinate of Bairro Alto are -25.4058225, -49.2076602.
The geographical coordinate of Barreirinha are -25.3685642, -49.2604545.
The geographical coordinate of Boa Vista are -25.4576591, -48.9292726.
The geographical coordinate of Cachoeira are -25.3539823, -49.2572706.
The geographical coordinate of Pilarzinho are -25.3963479, -49.2875575.
The geographical coordinate of Santa Cândida are -25.3698739, -49.2305741.
The geographical coordinate of São Lourenço are -25.3887613, -49.266281.
The geographical coordinate of Taboão are -25.3733813, -49.2807648.
The geographical coordinate of Tarumã are -25.4243559, -49.2221039.
The geographical coordinate of Tingui are -25.385354, -49.2240749.
The geographical coordinate of Alto Boqueirão are -25.532668, -49.2390629.
The geographical coordinate of Boqueirão are -25.500728, -49.2411054.
The geographical coordinate of Hauer are -25.4786086, -49.2534651.
The geographical coordinate of Xaxim are -25.5008748, -49.2678646.
The geographical coordinate of Cajuru are -25.4533496, -49.2130027.
The geographical coordinate of Capão da Imbuia are -25.4372102, -49.2120387.
The geographical coordinate of Guabirotuba are -25.4624958, -49.2433392.
The geographical coordinate of Jd. das Américas are -25.4536246, -49.2305768.
The geographical coordinate of Uberaba are -25.4849252, -49.2153133.
The geographical coordinate of Augusta are -25.4743152, -49.3714784.
The geographical coordinate of Cidade Industrial are -25.497993, -49.334522.
The geographical coordinate of Riviera are -25.4366101, -49.3813372.
The geographical coordinate of São Miguel are -25.5027409, -49.3610271.
The geographical coordinate of Água Verde are -25.4552633, -49.2828084.
The geographical coordinate of Campo Comprido are -25.4533397, -49.3284321.
The geographical coordinate of Fanny are -25.4791996, -49.2661377.
The geographical coordinate of Fazendinha are -25.8790212, -50.5419599.
The geographical coordinate of Guaíra are -25.4700425, -49.2752424.
The geographical coordinate of Lindoia are -25.479004, -49.2776917.
The geographical coordinate of Novo Mundo are -25.4869659, -49.2960629.
The geographical coordinate of Parolin are -25.4599763, -49.2637669.
The geographical coordinate of Portão are -25.4737001, -49.302414.
```

```
The geographical coordinate of Santa Quitéria are -25.462602, -49.3109435.
The geographical coordinate of Vila Izabel are -25.4563261, -49.2939793.
The geographical coordinate of Ahú are -25.3998414, -49.2619428.
The geographical coordinate of Alto da Glória are -25.4194476, -49.2621486.
The geographical coordinate of Alto da XV are -25.4280673, -49.2511966.
The geographical coordinate of Batel are -25.4387449, -49.287052.
The geographical coordinate of Bigorrilho are -25.4339089, -49.2994023.
The geographical coordinate of Bom Retiro are -25.4088016, -49.2775763.
The geographical coordinate of Cabral are -25.4074203, -49.2515182.
The geographical coordinate of Centro are -25.7765741, -49.3270131.
The geographical coordinate of Centro Cívico are -25.4143755, -49.2683269.
The geographical coordinate of Cristo Rei are -25.4336461, -49.2445794.
The geographical coordinate of Hugo Lange are -25.4199794, -49.2462298.
The geographical coordinate of Jardim Botânico are -25.441716900000003, -49.239
50308076781.
The geographical coordinate of Jardim Social are -25.4196583, -49.234537.
The geographical coordinate of Juvevê are -25.4160133, -49.2544262.
The geographical coordinate of Mercês are -25.4246085, -49.2905022.
The geographical coordinate of Prado Velho are -25.4533706, -49.2544113.
The geographical coordinate of Rebouças are -25.4444772, -49.2646681.
The geographical coordinate of São Francisco are -25.4237543, -49.2770883.
The geographical coordinate of Campo de Santana are -25.6003501, -49.33376.
The geographical coordinate of Capão Raso are -25.504632, -49.2985781.
The geographical coordinate of Caximba are -25.6208968, -49.347262.
The geographical coordinate of Pinheirinho are -25.5228792, -49.2904118.
The geographical coordinate of Tatuquara are -25.5624694, -49.3215965.
The geographical coordinate of Butiatuvinha are -25.4001938, -49.3562253.
The geographical coordinate of Campina do Siqueira are -25.4387255, -49.308722
2.
The geographical coordinate of Cascatinha are -25.4152363, -49.3105765.
The geographical coordinate of Lamenha Pequena are -25.3660645, -49.3386627.
The geographical coordinate of Mossunguê are -25.4347057, -49.330387.
The geographical coordinate of Orleans are -25.4284385, -49.3575305.
The geographical coordinate of Santa Felicidade are -25.4026959, -49.3285775.
The geographical coordinate of Santo Inácio are -25.4252057, -49.3285776.
The geographical coordinate of São Braz are -25.4182261, -49.350834.
The geographical coordinate of São João are -25.3914534, -49.3114794.
The geographical coordinate of Seminário are -25.4489103, -49.3051474.
The geographical coordinate of Vista Alegre are -25.4067665, -49.2955782.
```

Let's visualize the dataframe generated:

In [63]:
```
df_cwb_coord
```

Out[63]:

	Neighborhood	Latitude	Longitude
0	Ganchinho	-25.572076	-49.263667
1	Sitio Cercado	-25.542701	-49.269106
2	Umbará	-25.568169	-49.285699
3	Abranches	-25.361474	-49.272054
4	Atuba	-25.387500	-49.206606
5	Bacacheri	-25.396850	-49.234456
6	Bairro Alto	-25.405822	-49.207660
7	Barreirinha	-25.368564	-49.260455
8	Boa Vista	-25.388479	-49.243713
9	Cachoeira	-25.353982	-49.257271
10	Pilarzinho	-25.396348	-49.287557

And save it as a file:

In [64]:
```
# save curitiba_df to pickle

with open('df_cwb_coord.pkl', 'wb') as f:
        pickle.dump(df_cwb_coord, f)
```

In [17]:
```
# load curitiba_df from pickle

with open('df_cwb_coord.pkl', 'rb') as f:
        df_cwb_coord = pickle.load(f)
```

Now let's merge curitiba_df with df_cwb_coord to get the coordinates on our main neighborhood database:

In [65]:
```
curitiba_df_coord = curitiba_df.merge(df_cwb_coord, on='Neighborhood', how='left
```

The result was a database with 76 neighborhoods and 10 columns that looks like the following:

In [66]:
```
curitiba_df_coord
```

Out[66]:

	Neighborhood	Area	Men	Women	Total	Households	Avg. Income	Population Density	Latit
0	Ganchinho	11.20	3667	3658	7325	1921	767.35	654.017857	-25.572
1	Sitio Cercado	11.12	50631	51779	102410	27914	934.95	9209.532374	-25.542
2	Umbará	22.47	7280	7315	14595	17064	908.70	649.532710	-25.568
3	Abranches	4.32	5463	5702	11165	3154	1009.67	2584.490741	-25.361
4	Atuba	4.27	6156	6476	12632	3627	1211.60	2958.313817	-25.387
5	Bacacheri	6.98	10762	12344	23106	7107	3029.00	3310.315186	-25.396
6	Bairro Alto	7.02	20244	21789	42033	12071	1211.60	5987.606838	-25.405
7	Barreirinha	3.73	8079	8942	17021	5024	1272.18	4563.270777	-25.368
8	Boa Vista	5.14	13677	15714	29391	9212	1817.40	5718.093385	-25.388
9	Cachoeira	3.07	3811	3927	7738	2091	908.70	2520.521173	-25.353

Finally we can save this neighborhood data base into a file with Pickle:

In [67]:
```
# save curitiba_df_coord to pickle

with open('curitiba_df_coord.pkl', 'wb') as f:
        pickle.dump(curitiba_df_coord, f)
```

In [4]:
```
# load curitiba_df_coord from pickle

with open('curitiba_df_coord.pkl', 'rb') as f:
        curitiba_df_coord = pickle.load(f)
```

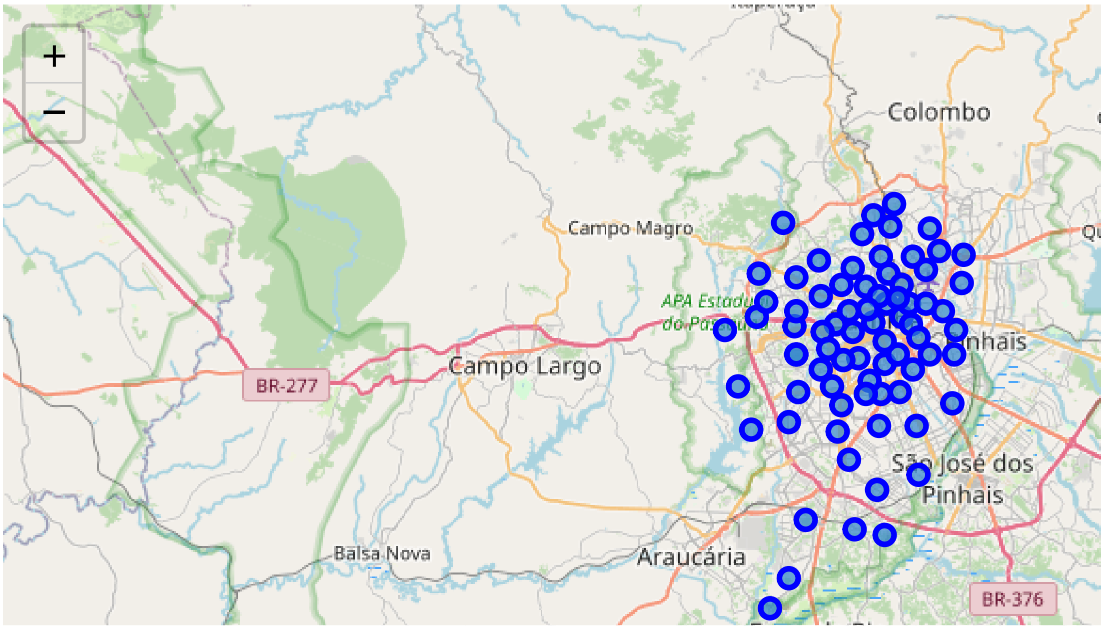
Let's visualize the data we have so far:

Curitiba neighborhood centers:

In [5]:

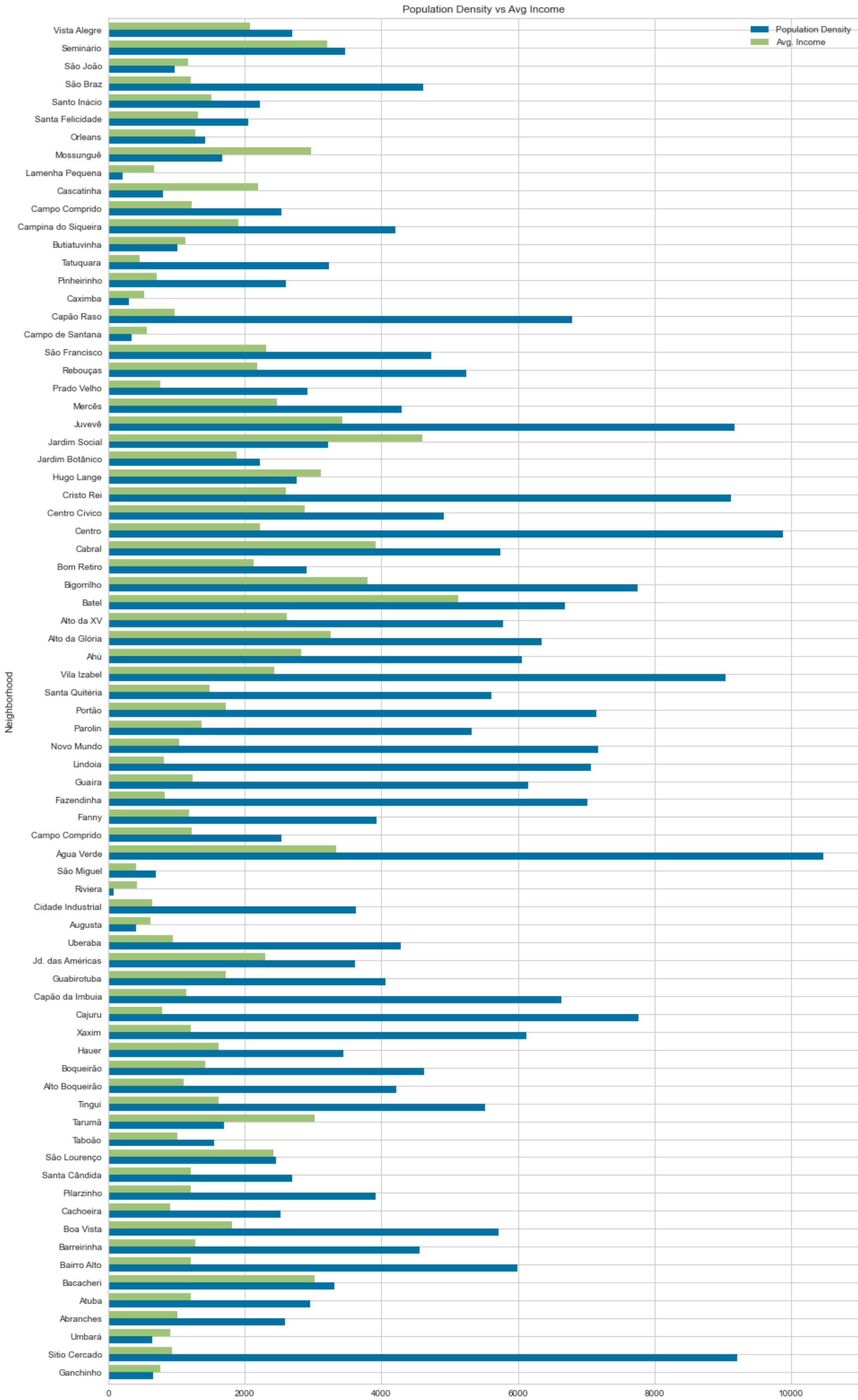
```
map_curitiba = folium.Map(location=[curitiba_df_coord['Latitude'][0], curitiba_d

# add markers to map
for lat, lng, neighborhood in zip(curitiba_df_coord['Latitude'],
                                   curitiba_df_coord['Longitude'],
                                   curitiba_df_coord['Neighborhood']):
    label = '{}'.format(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=0.7,
        parse_html=False).add_to(map_curitiba)

map_curitiba
```

Out[5]:



And to start understanding how the population is distributted:

In [29]:
```
# plot data


curitiba_df_coord[['Population Density', 'Avg. Income']].plot(kind='barh', figsi

#plt.xlabel('Borough') # add to x-label to the plot
plt.ylabel('Neighborhood') # add y-label to the plot
plt.title('Population Density vs Avg Income') # add title to the plot
plt.yticks (np.arange(len(curitiba_df_coord)), curitiba_df_coord['Neighborhood']
plt.show()
```



3st Data - Foursquare

Now that we have our location candidates, let's use Foursquare API to get info on restaurants in each neighborhood.

Foursquare credentials are defined in hidden cell bellow.

In [ ]:
```
## Define Foursquare Credentials and Version
CLIENT_ID = '' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
ACCESS_TOKEN = '' # your FourSquare Access Token
VERSION = '20210330'
LIMIT = 100

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

Define function to get nearby restaurants

In [10]:
```
def getNearbyRestaurants(names, latitudes, longitudes, radius):
    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?&section=food&client
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        resp = requests.get(url).json()["response"]

        if "groups" in resp:
            results = resp['groups'][0]['items']
            # 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']['id'],
                v['venue']['categories'][0]['name']) for v in results])

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

    return(nearby_venues)
```

Curitiba Restaurants within a 1000m radius from neighborhood coordinates:

In [11]:
```
curitiba_restaurants_1000 = getNearbyRestaurants(names=curitiba_df_coord['Neighb
                                                 latitudes=curitiba_df_coord['Latitude']
                                                 longitudes=curitiba_df_coord['Longitude
                                                 radius=1000)
```

Let's see how it looks:

In [39]:
```
curitiba_restaurants_1000.head(10)
```

Out[39]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	
0	Abranches	-25.361474	-49.272054	Panificadora E Confeitaria Espírito Santo	-25.356506	-49.273200	595aa015f427d
1	Abranches	-25.361474	-49.272054	D1a Lanches	-25.363250	-49.262613	50f806d4e4b0
2	Abranches	-25.361474	-49.272054	Casa de Carnes Quality	-25.365375	-49.271793	60439b4df488
3	Abranches	-25.361474	-49.272054	Famas Beer	-25.366502	-49.275230	525b4e3d11d2
4	Abranches	-25.361474	-49.272054	Panificadora E Confeitaria Hayama	-25.369311	-49.268146	4dbd83c30437
5	Ahú	-25.399841	-49.261943	Batataria Curitiba	-25.402340	-49.260083	537fe797498e2
6	Ahú	-25.399841	-49.261943	MaisQPão	-25.401814	-49.261891	4da8c39d8154
7	Ahú	-25.399841	-49.261943	Calenzano Pizzarias	-25.401579	-49.262915	4e18fdbde4cd
8	Ahú	-25.399841	-49.261943	Trigo & Cia	-25.406493	-49.260479	4d6fffc1b09a
9	Ahú	-25.399841	-49.261943	Hotdog Mada	-25.398675	-49.256338	4e727af0aeb7

And the size of this dataframe:

In [13]:
```
curitiba_restaurants_1000.shape
```

Out[13]: (3085, 8)

Considering a radius of 1000m, there may be an overlap between neighborhoods and consequently duplicated restaurants. It can be confirmed as follows:

In [14]:
```
print('{} uniques restaurants.'.format(len(curitiba_restaurants_1000['Venue ID']
```

```
2369 uniques restaurants.
```

To deal with that, let´s consider only the closest restaurant to the neighborhood center, and drop the other. The first thing to do is calculate the distance between each restaurant and the central coordinate of the neighborhood:

In [15]:
```
#Function to calculate the distance between 2 coordinates

wgs84_geod = Geod(ellps='WGS84') #Distance will be measured on this ellipsoid -

#Get distance between pairs of lat-lon points
def Distance(lat1,lon1,lat2,lon2):
  az12,az21,dist = wgs84_geod.inv(lon1,lat1,lon2,lat2) #Yes, this order is corre
  return dist
```

In [16]:
```
#Apply the function to our dataframe

curitiba_restaurants_1000['Dist'] = Distance(curitiba_restaurants_1000['Neighbor
                                             curitiba_restaurants_1000['Neig
                                             curitiba_restaurants_1000['Venu
                                             curitiba_restaurants_1000['Venu
```

In [17]:
```
#Sort in ascending order:

curitiba_restaurants_1000.sort_values(['Dist'], ascending=[True], inplace=True)
```

In [18]:
```
#Drop duplicates keeping first entry:

curitiba_restaurants_1000 =  curitiba_restaurants_1000.drop_duplicates(subset='V
```

In [ ]:
```
#Sort by neighborhood again:

curitiba_restaurants_1000.sort_values(['Neighborhood'], ascending=[True], inplac
```

Reset index after dropping the duplicates and now we can explore our final dataframe, with 2368 restaurants:

In [38]:
```
curitiba_restaurants_1000.reset_index(drop=True, inplace=True)
```

4st Data - Foursquare API Premium Calls

Next step is to obtain some statistics about the restaurants, to reach our final goal of this project. To do that, we will call upon Foursquare one more time, retrieving likes, rating and prices from our restaurants dataset:

In [21]:
```
#creating empty lists

like_list = []
tip_list = []
price_list = []
rating_list = []
```

In [22]:
```
# Define a function to parse JSON file

def json_get(dictionary, dot_path, default=None):
    path = dot_path.split('.')
    try:
        return reduce(dict.__getitem__, path, dictionary)
    except KeyError:
        return default
    except TypeError:
        return default
```

Our database has 2369 items, so we will split the request on 5 due to the Foursquare 500 daily API limit for premium calls.

In [40]:
```
# set up to pull the likes, rates and price from the API based on venue ID


url_list = []

for venue_id in list(curitiba_restaurants_1000['Venue ID'][1988:2400].tolist()):
    venue_url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_sec
    url_list.append(venue_url)

for link in url_list:
    #print(link)
    #print(result)
    result = requests.get(link).json()
    likes = json_get(result, 'response.venue.likes.count')
    tip = json_get(result, 'response.venue.stats.tipCount')
    price = json_get(result, 'response.venue.price.tier')
    rating = json_get(result, 'response.venue.rating')
    like_list.append(likes)
    tip_list.append(tip)
    price_list.append(price)
    rating_list.append(rating)
#nearby_venues['likes'] = like_list
#nearby_venues.head()
```

To avoid data loss during the five days of collections, we will save the partial results using Pickle one more time:

In [43]:
```
# save partial results to pickle, to continue in next day


with open('curitiba_restaurants_1000.pkl', 'wb') as f:
        pickle.dump(curitiba_restaurants_1000, f)
with open('like_list.pkl', 'wb') as f:
        pickle.dump(like_list, f)
with open('tip_list.pkl', 'wb') as f:
        pickle.dump(tip_list, f)
with open('price_list.pkl', 'wb') as f:
        pickle.dump(price_list, f)
with open('rating_list.pkl', 'wb') as f:
        pickle.dump(rating_list, f)
```

In [42]:
```
len(like_list)
```

Out[42]: 2369

In [34]:
```
# load from pickle

with open('curitiba_restaurants_1000.pkl', 'rb') as f:
        curitiba_restaurants_1000 = pickle.load(f)
with open('like_list.pkl', 'rb') as f:
        like_list = pickle.load(f)
with open('tip_list.pkl', 'rb') as f:
        tip_list = pickle.load(f)
with open('price_list.pkl', 'rb') as f:
        price_list = pickle.load(f)
with open('rating_list.pkl', 'rb') as f:
        rating_list = pickle.load(f)
```

Now lets join the results into our main restaurants dataframe and name it curitiba_restaurants, that is our final database to be explored.

In [ ]:
```
curitiba_restaurants = curitiba_restaurants_1000
curitiba_restaurants['Likes'] = like_list
curitiba_restaurants['Tips'] = tip_list
curitiba_restaurants['Price'] = price_list
curitiba_restaurants['Rating'] = rating_list
```

In [46]:
```
# save to pickle curitiba_restaurants

with open('curitiba_restaurants.pkl', 'wb') as f:
        pickle.dump(curitiba_restaurants, f)
```

In [7]:
```
# load from pickle curitiba_restaurants

with open('curitiba_restaurants.pkl', 'rb') as f:
        curitiba_restaurants = pickle.load(f)
```

Considering the goal of our project, only evaluated restaurants must be in our final database and after dropping rows with NaN in the columns of interest (ratings, likes, tips and price), 1268 restaurants remained for our analysis, as follows:

In [8]:
```
curitiba_restaurants = curitiba_restaurants.dropna()
```

In [9]:
```
curitiba_restaurants.shape
```

Out[9]: (1268, 13)

In [10]:
```
curitiba_restaurants.reset_index(drop=True, inplace=True)
```

Quickly examine the resulting dataframe.

In [11]:
```
curitiba_restaurants.head(10)
```

Out[11]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	
0	Ahú	-25.399841	-49.261943	Batataria Curitiba	-25.402340	-49.260083	537fe7974
1	Ahú	-25.399841	-49.261943	Calenzano Pizzarias	-25.401579	-49.262915	4e18fdbd
2	Ahú	-25.399841	-49.261943	Trigo & Cia	-25.406493	-49.260479	4d6fffc1l
3	Ahú	-25.399841	-49.261943	Restaurante Chaminé	-25.402534	-49.269136	4cb48ce77
4	Ahú	-25.399841	-49.261943	Albatroz	-25.401421	-49.271737	4bb8bad1
5	Ahú	-25.399841	-49.261943	Subway	-25.403057	-49.258522	5436a8b2
6	Ahú	-25.399841	-49.261943	Kenji Kaiten	-25.406707	-49.265539	513376b2e

Methodology

In this project we will direct our efforts on detecting areas of Curitiba that have high restaurant density, particularly those with high number of cheap and well evaluated restaurants, resulting in an interactive map of Where to Eat Well fot Cheap in Curitiba.

In first step we have collected the required **data: location and type (category) of every restaurant within 1km from each neighborhood center**. We have also collected **Foursquarer user´s evaluations** of those restaurants (API premium calls).

Second step in our analysis will be the exploration of Curitiba, in terms of **restaurants distribution, evaluation and price statistics and top categories** per neighborhood.

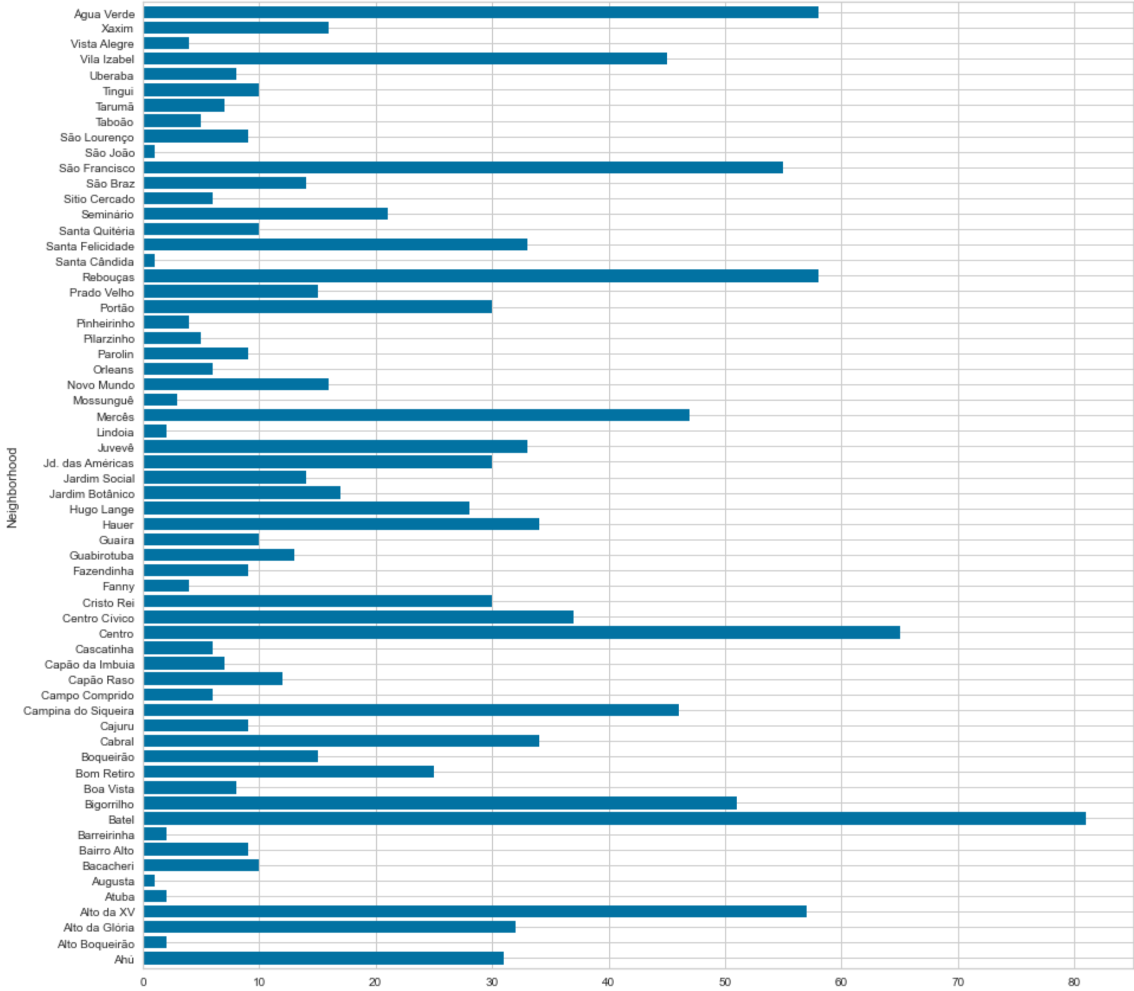
In third and final step we will create cluster (using **k-means clustering**) of restaurants based on **price and rating**, identify the cluster with higher rates and lower price and focus on this list to create interactive maps with the results.

Analysis

Number of Restaurants per Neighborhood

Let's start exploring the neighborhoods counting the number of restaurants on each neighborhood and visualize in a graphic:

In [212]:
```
# Plot data

curitiba_restaurants_count = curitiba_restaurants.groupby('Neighborhood').count(
curitiba_restaurants_count['Venue'].plot(kind='barh', figsize=(15, 15), width =
plt.show()
```



Another information that might be usefull for our analisys to understand the city of Curitiba, is getting the number of restaurants by Category, ranking and taking a look into the 20 most common:

In [234]:
```
curitiba_rest_cat_count = curitiba_restaurants.groupby(['Venue Category']).count
curitiba_rest_cat_count.drop(['Neighborhood Latitude', 'Neighborhood Longitude',
curitiba_rest_cat_count.columns = ['Number of restaurants']
curitiba_rest_cat_count.sort_values(by='Number of restaurants', ascending=False,
curitiba_rest_cat_count.head(20)
```

Out[234]:

Venue Category	Number of restaurants
Pizza Place	146
Brazilian Restaurant	146
Bakery	129
Restaurant	89
Italian Restaurant	88
Café	74
Burger Joint	59
Fast Food Restaurant	44
BBQ Joint	44
Hot Dog Joint	37
Sandwich Place	36
Middle Eastern Restaurant	31
Chinese Restaurant	28
Japanese Restaurant	28
Snack Place	27
Steakhouse	24
Food Truck	24
Seafood Restaurant	24
Gastropub	19
Asian Restaurant	15

And some statistics grouped by Neighborhood about the likes, rating and prices obtained from Foursquare:

In [45]:

```
cwb_rest_neigh_stats = curitiba_restaurants.groupby('Neighborhood').agg({'Likes'
cwb_rest_neigh_stats.columns = ['Likes_Sum', 'Tips_Sum', 'Rating_Mean', 'Price_M
cwb_rest_neigh_stats.sort_values(by='Likes_Sum', ascending=False, inplace=True)
cwb_rest_neigh_stats.head(20)
```

Out[45]:

	Likes_Sum	Tips_Sum	Rating_Mean	Price_Mean
Neighborhood				
Batel	12701	4037	7.835802	2.012346
Centro	7620	3047	7.840000	1.769231
Santa Felicidade	6431	1806	6.721212	1.848485
Cabral	5889	2057	7.017647	1.911765
Água Verde	4898	2138	7.162069	1.775862
Campina do Siqueira	4871	1432	6.578261	1.826087
Rebouças	4508	1619	6.544828	1.620690
Vila Izabel	4299	1561	7.193333	1.644444
Hugo Lange	3817	1270	7.382143	2.000000
Juvevê	3538	1224	7.278788	1.787879
Seminário	3390	865	7.114286	1.761905
São Francisco	3338	1165	7.520000	1.709091
Mercês	3240	1214	7.021277	1.914894
Bigorrilho	3199	1300	7.054902	1.666667
Alto da XV	3139	1232	6.715789	1.701754
Portão	2932	893	6.806667	1.666667
Cristo Rei	2735	1071	6.856667	2.000000
Centro Cívico	2396	997	6.735135	1.891892
Bom Retiro	2053	659	7.012000	2.040000
Jd. das Américas	1511	542	6.666667	1.500000

Neighborhood´s Top Categories

Type *Markdown* and LaTeX: α^2

In [47]:

```
# one hot encoding
cwb_onehot = pd.get_dummies(curitiba_restaurants[['Venue Category']], prefix="",

# add neighborhood column back to dataframe
cwb_onehot['Neighborhood'] = curitiba_restaurants['Neighborhood']

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

print(cwb_onehot.shape)
cwb_onehot.head()
```

(1268, 59)

Out[47]:

	Neighborhood	American Restaurant	Argentinian Restaurant	Asian Restaurant	BBQ Joint	Bakery	Brazilian Restaurant	Breakfast Spot	Buffe
0	Ahú	0	0	0	0	0	0	0	(
1	Ahú	0	0	0	0	0	0	0	(
2	Ahú	0	0	0	0	1	0	0	(
3	Ahú	0	0	0	0	0	1	0	(
4	Ahú	0	0	0	0	0	0	0	(

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [48]:

```
cwb_grouped = cwb_onehot.groupby('Neighborhood').mean().reset_index()
cwb_grouped
```

Out[48]:

	Neighborhood	American Restaurant	Argentinian Restaurant	Asian Restaurant	BBQ Joint	Bakery	Brazilian Restaurant	Break S
0	Ahú	0.000000	0.000000	0.000000	0.032258	0.096774	0.096774	0.000
1	Alto Boqueirão	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000
2	Alto da Glória	0.000000	0.000000	0.062500	0.000000	0.062500	0.187500	0.000
3	Alto da XV	0.000000	0.017544	0.000000	0.052632	0.070175	0.140351	0.000
4	Atuba	0.000000	0.000000	0.000000	0.000000	0.500000	0.000000	0.000
5	Augusta	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
6	Bacacheri	0.000000	0.000000	0.000000	0.000000	0.300000	0.100000	0.000
7	Bairro Alto	0.000000	0.000000	0.000000	0.222222	0.000000	0.000000	0.000
8	Barreirinha	0.000000	0.000000	0.000000	0.000000	0.500000	0.500000	0.000

Lets write a function to sort the venues in descending order.

In [49]:

```
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.

In [51]:

```
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'] = cwb_grouped['Neighborhood']

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

neighborhoods_venues_sorted
```

Out[51]:

	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	7 C
0	Ahú	Pizza Place	Seafood Restaurant	Restaurant	Bakery	Brazilian Restaurant	Café	Re
1	Alto Boqueirão	Bakery	American Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	Molecular Gastronomy Restaurant	A Re
2	Alto da Glória	Brazilian Restaurant	Café	Snack Place	Chinese Restaurant	Asian Restaurant	Bakery	Re
3	Alto da XV	Brazilian Restaurant	Restaurant	Café	Pizza Place	Bakery	Italian Restaurant	J Re
4	Atuba	Bakery	Café	American Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	M Gas Re
5	Augusta	Snack Place	American Restaurant	Sandwich Place	Mexican Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	M Gas Re
6	Bacacheri	Hot Dog Joint	Bakery	Brazilian Restaurant	Pizza Place	Café	Sandwich Place	A Re
7	Bairro Alto	Pizza Place	BBQ Joint	Sandwich Place	Tapiocaria	Hot Dog Joint	Restaurant	A Re
8	Barreirinha	Bakery	Brazilian Restaurant	American Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	M Gas Re
9	Batel	Italian Restaurant	Café	Pizza Place	Brazilian Restaurant	Burger Joint	Gastropub	Re
10	Bigorrilho	Pizza Place	Bakery	Italian Restaurant	Brazilian Restaurant	Chinese Restaurant	BBQ Joint	

	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	7 C
11	Boa Vista	Burger Joint	Food Truck	Pizza Place	Café	Deli / Bodega	Salad Place	Re
12	Bom Retiro	Steakhouse	Restaurant	BBQ Joint	Bakery	Brazilian Restaurant	New American Restaurant	Re
13	Boqueirão	Bakery	Pizza Place	Hot Dog Joint	Café	Burger Joint	Food Truck	
14	Cabral	Pizza Place	Brazilian Restaurant	Restaurant	Bakery	Café	Diner	
15	Cajuru	Bakery	Pizza Place	Sandwich Place	Hot Dog Joint	Restaurant	American Restaurant	Re
16	Campina do Siqueira	Pizza Place	Fast Food Restaurant	Italian Restaurant	Brazilian Restaurant	Café	Restaurant	Ste
17	Campo Comprido	BBQ Joint	Bakery	Middle Eastern Restaurant	Restaurant	American Restaurant	Seafood Restaurant	Re
18	Capão Raso	Pizza Place	Bakery	Sandwich Place	Food Truck	BBQ Joint	Brazilian Restaurant	
19	Capão da Imbuia	Bakery	Brazilian Restaurant	Burger Joint	BBQ Joint	Restaurant	Pastelaria	Sal
20	Cascatinha	Italian Restaurant	Pizza Place	Churrascaria	American Restaurant	Sandwich Place	Middle Eastern Restaurant	Re
21	Centro	Brazilian Restaurant	Middle Eastern Restaurant	Café	Restaurant	Snack Place	Burger Joint	Re
22	Centro Cívico	Brazilian Restaurant	Café	Italian Restaurant	Restaurant	Asian Restaurant	Pizza Place	/ Re
23	Cristo Rei	Restaurant	Pizza Place	Bakery	Brazilian Restaurant	Italian Restaurant	BBQ Joint	Bur
24	Fanny	Bakery	Hot Dog Joint	Fast Food Restaurant	American Restaurant	Sandwich Place	Mineiro Restaurant	M Gas Re
25	Fazendinha	Pizza Place	Diner	Bakery	Fried Chicken Joint	Fast Food Restaurant	Steakhouse	/ Re
26	Guabirotuba	Café	Sandwich Place	Bakery	Wings Joint	Food Truck	Sushi Restaurant	Chu
27	Guaíra	Brazilian Restaurant	Pizza Place	Café	Japanese Restaurant	Bakery	Diner	/ Re
28	Hauer	Fast Food Restaurant	BBQ Joint	Brazilian Restaurant	Pizza Place	Café	Sandwich Place	
29	Hugo Lange	Restaurant	Pizza Place	Brazilian Restaurant	Bakery	Middle Eastern Restaurant	Burger Joint	Re
30	Jardim Botânico	Brazilian Restaurant	Burger Joint	Fast Food Restaurant	Chinese Restaurant	Vegetarian / Vegan Restaurant	Bakery	
31	Jardim Social	Bakery	Italian Restaurant	Soup Place	Hot Dog Joint	Fast Food Restaurant	BBQ Joint	Re
32	Jd. das Américas	Brazilian Restaurant	Pizza Place	Bakery	Fast Food Restaurant	Italian Restaurant	Churrascaria	Bur

	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	7 C
33	Juvevê	Pizza Place	Brazilian Restaurant	Italian Restaurant	Bakery	Restaurant	BBQ Joint	Bur
34	Lindoia	Hot Dog Joint	Restaurant	American Restaurant	Sandwich Place	Middle Eastern Restaurant	Mineiro Restaurant	M Gas Re
35	Mercês	Pizza Place	Brazilian Restaurant	Middle Eastern Restaurant	Bakery	Italian Restaurant	Burger Joint	
36	Mossunguê	Bakery	Pizza Place	American Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	M Gas Re
37	Novo Mundo	Fast Food Restaurant	Pizza Place	Brazilian Restaurant	Snack Place	Italian Restaurant	Burger Joint	
38	Orleans	Snack Place	BBQ Joint	Brazilian Restaurant	Steakhouse	Pizza Place	Restaurant	S
39	Parolin	BBQ Joint	Snack Place	Bakery	Brazilian Restaurant	Restaurant	Italian Restaurant	S
40	Pilarzinho	Pizza Place	Hot Dog Joint	Restaurant	Snack Place	American Restaurant	Sandwich Place	Re
41	Pinheirinho	BBQ Joint	Brazilian Restaurant	Burger Joint	American Restaurant	Seafood Restaurant	Mineiro Restaurant	M Gas Re
42	Portão	Bakery	Pizza Place	Fast Food Restaurant	Sandwich Place	Brazilian Restaurant	Burger Joint	Re
43	Prado Velho	Brazilian Restaurant	Café	Restaurant	Sandwich Place	Churrascaria	Bakery	Re
44	Rebouças	Brazilian Restaurant	Bakery	Burger Joint	Pizza Place	Restaurant	Chinese Restaurant	Re
45	Santa Cândida	Fast Food Restaurant	American Restaurant	Sandwich Place	Middle Eastern Restaurant	Mineiro Restaurant	Molecular Gastronomy Restaurant	A Re
46	Santa Felicidade	Italian Restaurant	Bakery	Seafood Restaurant	Fast Food Restaurant	Café	Snack Place	Re
47	Santa Quitéria	Bakery	Burger Joint	BBQ Joint	Restaurant	Pizza Place	Gastropub	Re
48	Seminário	Pizza Place	Brazilian Restaurant	Buffet	Italian Restaurant	Sandwich Place	Fried Chicken Joint	Sna
49	Sitio Cercado	Fast Food Restaurant	Bakery	Hot Dog Joint	Burger Joint	Pizza Place	Sandwich Place	A Re
50	São Braz	Bakery	Pizza Place	Italian Restaurant	Hot Dog Joint	Fast Food Restaurant	Sandwich Place	J Re
51	São Francisco	Brazilian Restaurant	Restaurant	Burger Joint	Café	Steakhouse	Pizza Place	Re
52	São João	Bakery	American Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Mineiro Restaurant	Molecular Gastronomy Restaurant	A Re
53	São Lourenço	Brazilian Restaurant	Italian Restaurant	BBQ Joint	Bakery	Japanese Restaurant	Pizza Place	

	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	7 C
54	Taboão	Snack Place	Bakery	Burger Joint	Restaurant	Seafood Restaurant	Middle Eastern Restaurant	Re
55	Tarumã	Fast Food Restaurant	Snack Place	Brazilian Restaurant	Burger Joint	Pizza Place	Sandwich Place	Re
56	Tingui	Brazilian Restaurant	Burger Joint	Fast Food Restaurant	Sandwich Place	Restaurant	Pizza Place	Re
57	Uberaba	Bakery	Food Truck	Hot Dog Joint	German Restaurant	Chinese Restaurant	Pizza Place	A Re
58	Vila Izabel	Pizza Place	Bakery	Japanese Restaurant	Brazilian Restaurant	Chinese Restaurant	Fast Food Restaurant	
59	Vista Alegre	Bakery	BBQ Joint	Brazilian Restaurant	American Restaurant	Seafood Restaurant	Mineiro Restaurant	M Gas Re
60	Xaxim	Pizza Place	Fast Food Restaurant	Burger Joint	Bakery	Food Truck	BBQ Joint	Re
61	Água Verde	Bakery	Brazilian Restaurant	Pizza Place	Restaurant	Burger Joint	Hot Dog Joint	Re

K-Means Clustering

Let us now **cluster** those locations to create **groups of restaurants with similar prices and rating**. After defining groups, we will try to identify the group with **the best ratings and lowest prices**.

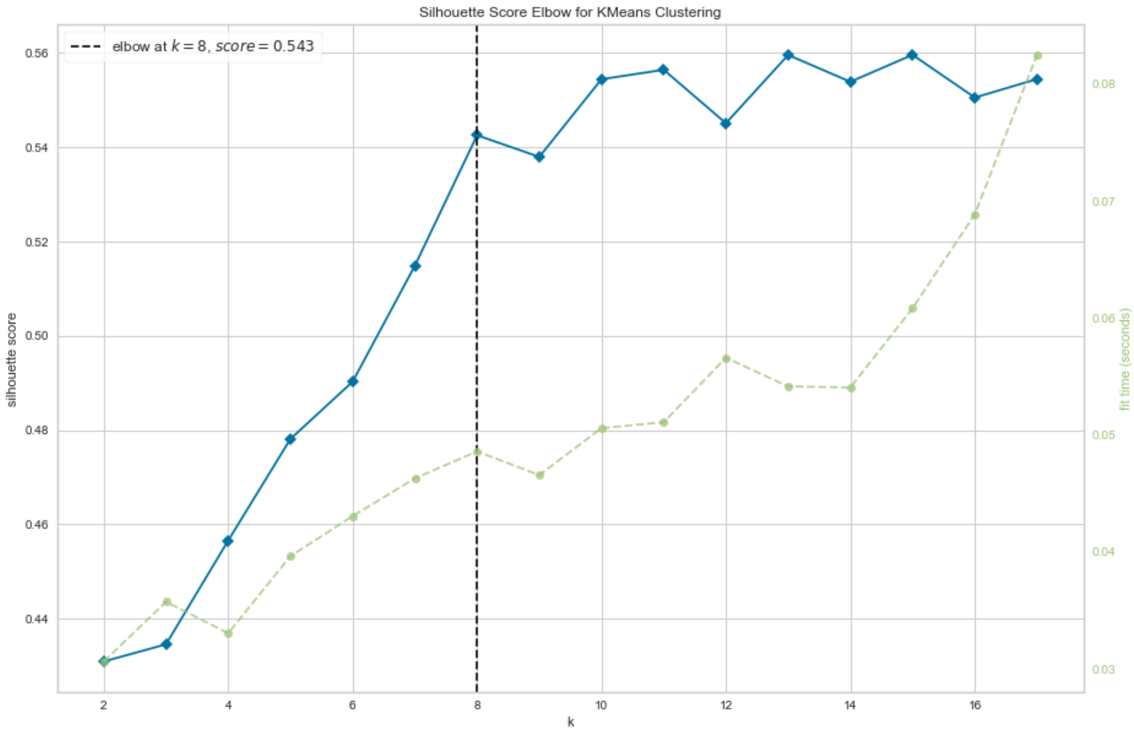
In [12]:
```
curitiba_rest_clustering = curitiba_restaurants.filter(['Price','Rating'])
curitiba_rest_clustering
```

Out[12]:

	Price	Rating
0	2.0	7.8
1	1.0	6.0
2	1.0	5.9
3	2.0	6.3
4	3.0	5.7
5	1.0	6.6
6	3.0	6.4
7	2.0	6.2
8	3.0	8.0
9	1.0	6.0
10	1.0	7.3

Identity the optimal number of clusters using KElbowVisualizer

In [54]:
```
# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(2,18), metric='silhouette', size=(1080,

visualizer.fit(curitiba_rest_clustering) # Fit the data to the visualizer
visualizer.poof() # Draw/show/poof the dat'
```



Out[54]: <AxesSubplot:title={'center':'Silhouette Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='silhouette score'>

Run k-means clustering with 8 clusters

In [13]:
```
# set number of clusters
kclusters = 8


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

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

Out[13]: array([4, 3, 3, ..., 5, 2, 2])

Join results with curitiba_restaurants dataframe

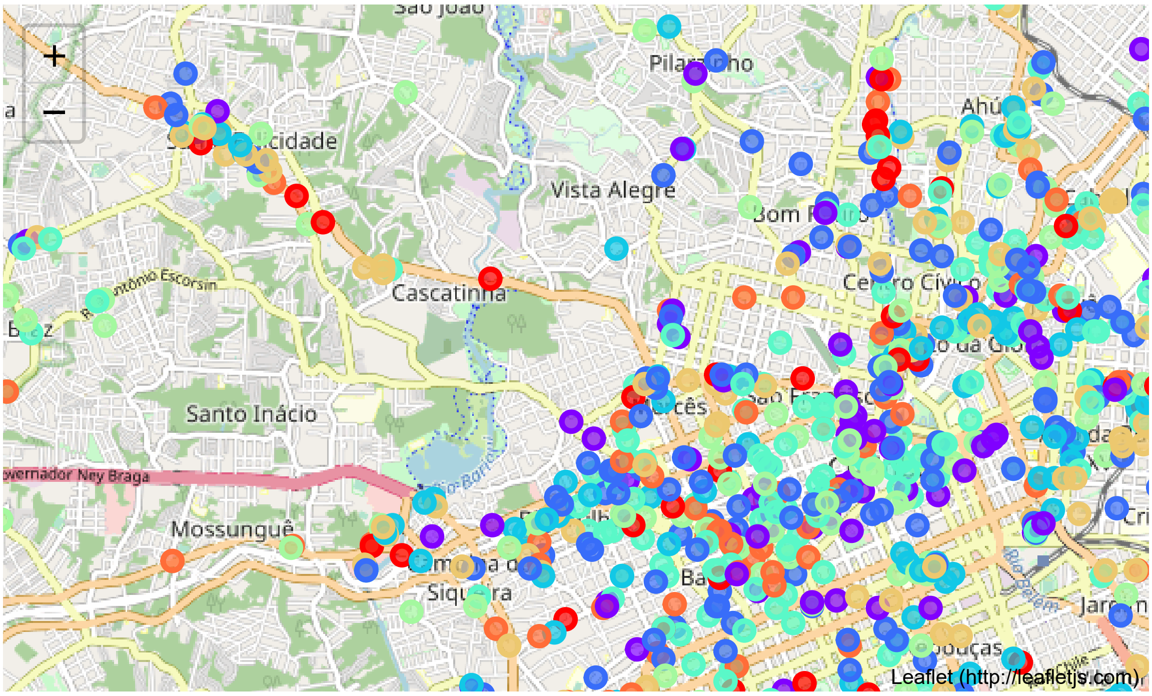
In [14]:
```
curitiba_restaurants['Cluster'] = kmeans.labels_
```

Let´s maps the results:

In [16]:

```
map_clusters = folium.Map(location=[curitiba_restaurants['Neighborhood Latitude'

# 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(curitiba_restaurants['Venue Latitude'], curiti
    label = folium.Popup(str(poi) + ' - Cluster ' + str(cluster))
    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
```

Out[16]:



The above map does not help much on for our analisys and to move on, we will identify the cluster of interest in the dataframe bellow.

In [17]:
```
curitiba_restaurants.head(80)
```

;	Ahú	-25.399841	-49.261943	Tempero do Titio	-25.399552	-49.271395	4d7cf3ca136bf04d5011688d	Ste
;	Ahú	-25.399841	-49.261943	Panificadora e Confeitaria A Massa	-25.401556	-49.258571	4dd812f4ae60680f15177e49	
'	Ahú	-25.399841	-49.261943	King Crab	-25.401180	-49.271536	4ef10b9a722ef86cc55f2a33	Re
;	Ahú	-25.399841	-49.261943	Predileta	-25.397524	-49.271365	4de95d0fd4c0faa56447d910	Piz
)	Ahú	-25.399841	-49.261943	Xi Wei Xian	-25.401325	-49.258512	5738b0eecd1055b3452a4470	Re
)	Ahú	-25.399841	-49.261943	Rei do Camarão	-25.397654	-49.271066	4c30bc97a0ced13a0b90126e	Re
	Alto Boqueirão	-25.532668	-49.239063	Panificadora Big Pão II	-25.531049	-49.244617	4eb10c905c5c80159c7c814c	
:	Alto Boqueirão	-25.532668	-49.239063	Panificadora Sartori	-25.536564	-49.242148	503a807fe4b0975cc34949a9	

We have identified as cluster number 2 and created a new dataframe with only this restaurants, what we called cwb_goodandcheap_rest.

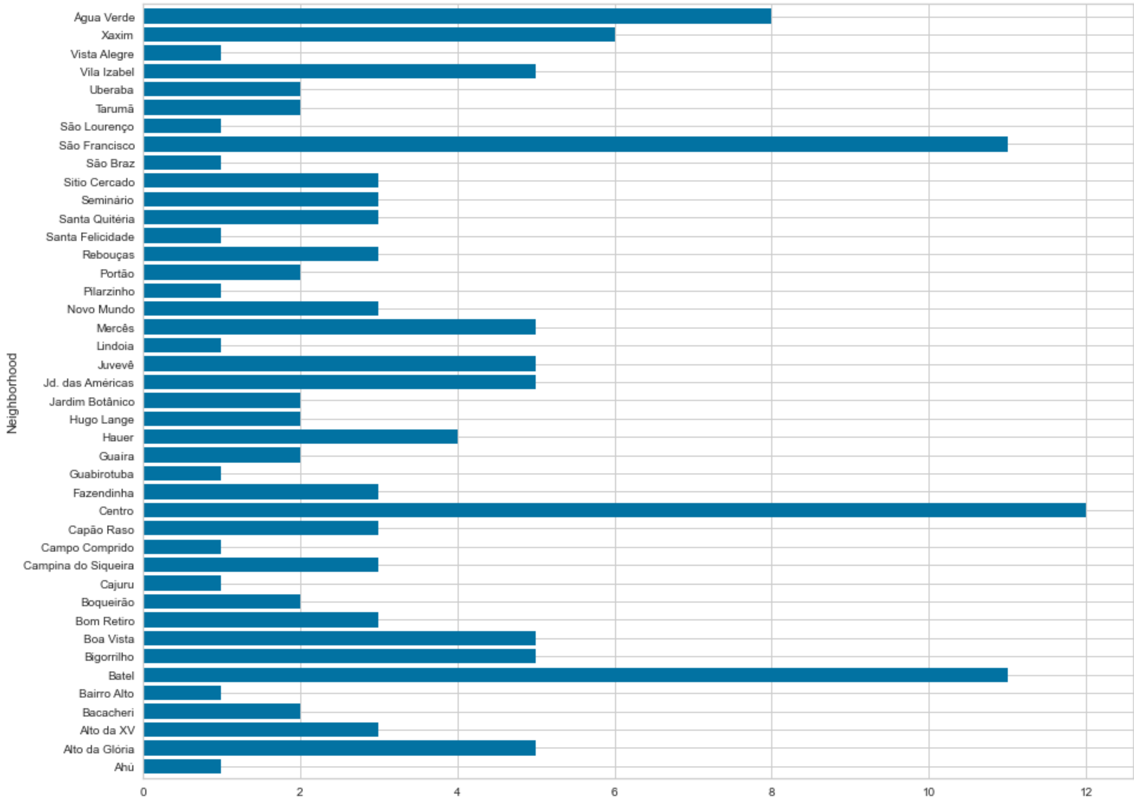
In [18]:
```
cluster = curitiba_restaurants['Cluster']==1
cwb_goodandcheap_rest = curitiba_restaurants[cluster]
cwb_goodandcheap_rest.reset_index(drop=True, inplace=True)
cwb_goodandcheap_rest.head()
```

Out[18]:

'hood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue ID	C
Ahú	-25.399841	-49.261943	Predileta	-25.397524	-49.271365	4de95d0fd4c0faa56447d910	
Glória	-25.419448	-49.262149	Cantina Açores	-25.415675	-49.263331	4c545fe5fd2ea59314abff29	Po Re
Glória	-25.419448	-49.262149	Red Velvet Coffee Shop	-25.426929	-49.260704	568ef6ca498e8e46ac1399c1	
Glória	-25.419448	-49.262149	Hot Dog Benassi	-25.415452	-49.259837	513529fae4b06f715c605c89	Fo
Glória	-25.419448	-49.262149	Casa da Coxinha	-25.427923	-49.261904	4c6c5ef7e13db60ca0e6d5b1	

Number of Good and Cheap Restaurants per Neighborhood

In [19]:
```
# Plot data

cwb_goodandcheap_rest_count = cwb_goodandcheap_rest.groupby('Neighborhood').coun
cwb_goodandcheap_rest_count['Venue'].plot(kind='barh', figsize=(15, 12), width =
plt.show()
```



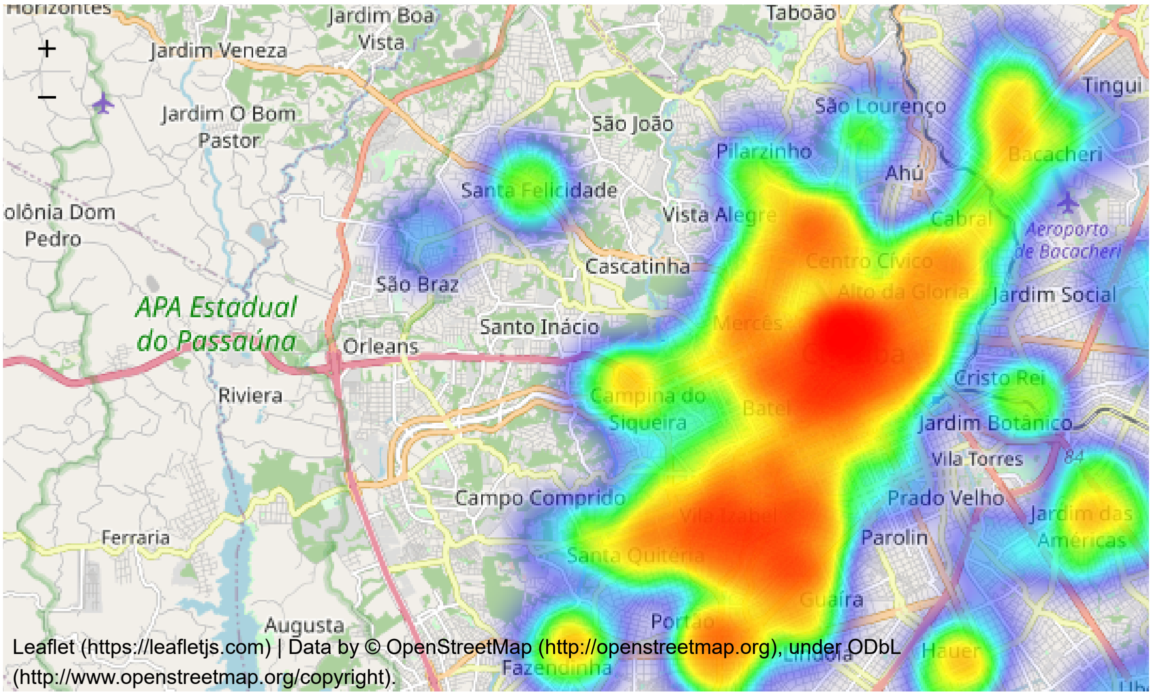
And a **heatmap** identify regions with the highest concentration of good and cheap restaurants.

In [59]:
```
cwb_heatdata_df = pd.DataFrame(columns = ['Venue Latitude','Venue Longitude'])

cwb_goodandcheap_rest_coord = cwb_goodandcheap_rest[['Venue Latitude','Venue Lon
cwb_heatdata_df['Venue Latitude'] = cwb_goodandcheap_rest_coord['Venue Latitude'
cwb_heatdata_df['Venue Longitude'] = cwb_goodandcheap_rest_coord['Venue Longitud

cwb_heatdata = []
cwb_heatdata = [[row['Venue Latitude'],row['Venue Longitude']] for index, row in
```

In [61]:

```
from folium.plugins import HeatMap


map_cwb_heatdata = folium.Map(location=[curitiba_df_coord['Latitude'][0], curiti
HeatMap(cwb_heatdata).add_to(map_cwb_heatdata)
map_cwb_heatdata
```

Out[61]:



To finish our analysis per neighborhood, some statistics:

In [65]:

```
cluster2df_stats = cwb_goodandcheap_rest.groupby('Neighborhood').agg({'Likes':'s
cluster2df_stats.columns = ['Likes_Sum', 'Tips_Sum', 'Rating_Mean']
cluster2df_stats.sort_values(by='Likes_Sum', ascending=False, inplace=True)
cluster2df_stats
```

Out[65]:

Neighborhood	Likes_Sum	Tips_Sum	Rating_Mean
Batel	958	227	7.960000
São Francisco	934	269	7.933333
Centro	875	316	8.250000
Água Verde	418	175	8.325000
Alto da Glória	384	159	7.850000
Hugo Lange	306	92	8.400000
Alto da XV	175	59	8.100000
Vila Izabel	169	64	7.900000
Jd. das Américas	143	70	8.060000
Xaxim	142	39	8.114286
Vista Alegre	142	68	8.700000
Bigorrilho	128	71	8.280000
Rebouças	118	44	7.933333
Santa Quitéria	115	32	7.766667
Juvevê	104	45	7.860000
Bacacheri	103	29	8.000000
Seminário	92	37	7.925000
Mercês	90	45	7.960000
Cabral	79	50	7.500000
Capão Raso	69	32	7.966667
Sitio Cercado	61	20	8.233333
Boa Vista	59	27	7.980000
Hauer	58	38	8.000000
Guaíra	55	15	8.150000
Uberaba	55	19	8.100000
Novo Mundo	53	22	8.266667
Bom Retiro	51	20	7.925000
Campina do Siqueira	48	19	7.833333
Guabirotuba	46	11	7.600000
Tarumã	42	12	8.100000
Portão	40	12	8.450000
Fazendinha	35	19	7.933333
Boqueirão	34	16	7.866667
Jardim Botânico	23	9	8.450000
Bairro Alto	22	6	8.500000
Santa Felicidade	20	13	8.000000

Neighborhood	Likes_Sum	Tips_Sum	Rating_Mean
São Braz	17	3	8.300000
Lindoia	17	7	8.000000
São Lourenço	15	9	8.000000
Cajuru	11	7	8.000000
Campo Comprido	10	3	7.900000
Pilarzinho	6	5	8.100000
Ahú	5	6	8.300000

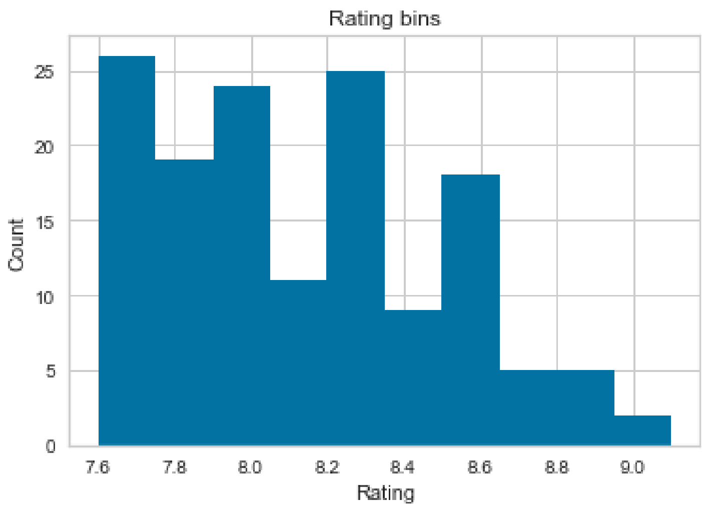
Binning Ratings

To go deeper in our analysis and make our interactive map more usefull, we can split the ratings in Low, Mid and High and plot them with different colors.

In [20]:

```
%matplotlib inline
plt.hist(cwb_goodandcheap_rest["Rating"])

# set x/y labels and plot title
plt.xlabel("Rating")
plt.ylabel("Count")
plt.title("Rating bins")
```

Out[20]: `Text(0.5, 1.0, 'Rating bins')`



We build a bin array, with a minimum value to a maximum value, with bandwidth calculated above. The bins will be values used to determine when one bin ends and another begins.

In [21]:

```
bins = np.linspace(min(cwb_goodandcheap_rest["Rating"]), max(cwb_goodandcheap_re

bins
```

Out[21]: `array([7.6, 8.1, 8.6, 9.1])`

We set group names:

In [22]:
```
group_names = ['Low', 'Medium', 'High']
```

We apply the function "cut" the determine what each value of "cwb_goodandcheap_rest["Rating"]" belongs to.

In [ ]:
```
cwb_goodandcheap_rest["Rating-Binned"] = pd.cut(cwb_goodandcheap_rest["Rating"],
cwb_goodandcheap_rest
```

We create a function do define colors for eath rating bin:

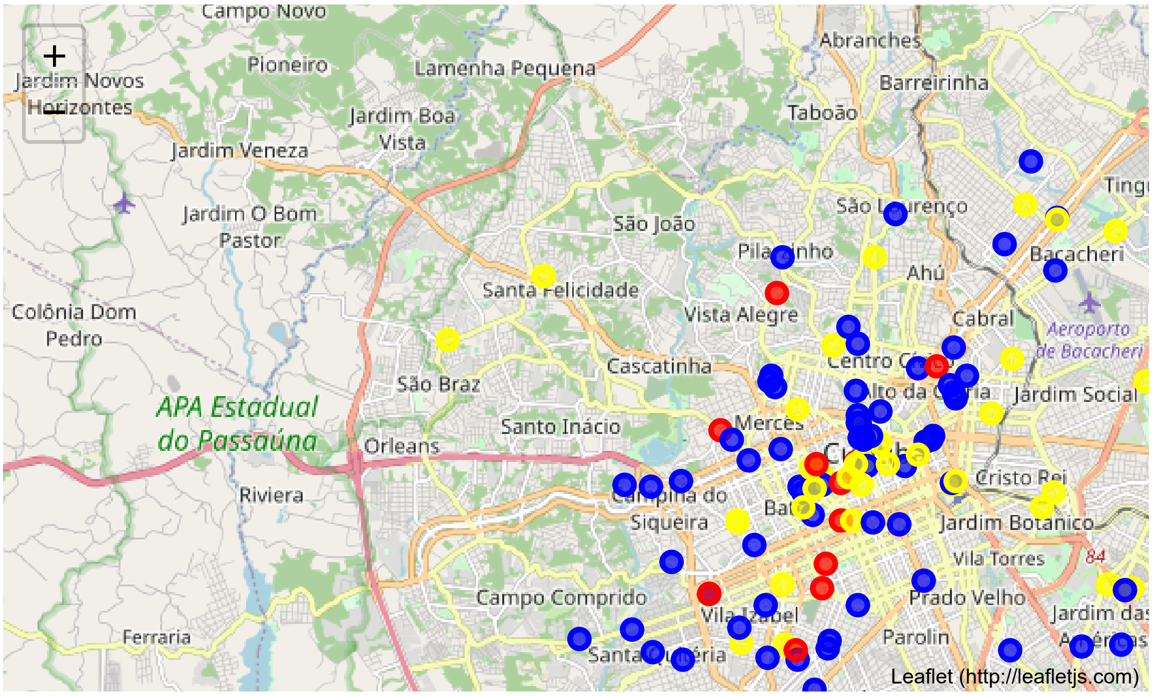
In [24]:
```
# Function to set color scheme for ratings
def color(argument):
    switcher = {
        'Low': 'blue',
        'Medium': 'yellow',
        'High': 'red',
    }
    return switcher.get(argument, "nothing")
```

Finaly, let's **create our interactive map** which can be used as a guide of **good and cheap restaurants in Curitiba**.

In [25]:

```
map_clusters = folium.Map(location=[curitiba_restaurants['Neighborhood Latitude'

# add markers to map
#markers_colors = []
for lat, lon, ven, rat, rbin in zip(cwb_goodandcheap_rest['Venue Latitude'], cwb
    label = folium.Popup(str(ven) + ' - Rating ' + str(rat), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=color(rbin),
        fill=True,
        fill_color=color(rbin),
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[25]:



Conclusion

Purpose of this project was to create a Curitiba guide of where to Eat for Cheap. During the analysis, several important statistical features of the boroughs and the restaurants were explored and visualized. Furthermore, clustering helped to idenitidy and highlight the group of restaurants we are looking for, and with the Heatmap and some statistics we have identified the region between the three neighborhoods of Centro, Batel and São Francisco as being of greatest interest for our goal. To refine our proposal, we have identified the possibility to rank the restaurants from this cluster based on their rating, to finally create the interactive map, our final objective.